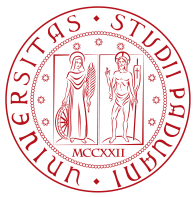


Università degli Studi di Padova

Department of Information Engineering

Ph.D. Course in Information Engineering

Science and Information Technology Curriculum

XXXIV Series

Visual Understanding across Multiple Semantic Groups, Domains and Devices

Coordinator

Prof. Andrea Neviani

Supervisor

Prof. Pietro Zanuttigh

Ph.D. Candidate

Umberto Michieli

Academic Year 2021/2022 (800^{th})

Abstract

Automatic pixel-level semantic scene understanding is a relevant problem in computer vision and a requirement for many practical applications such as autonomous vehicles, robotic navigation systems, video surveillance and medical tasks. Usually, semantic segmentation is performed with deep learning architectures. Unfortunately, deep neural networks on practical benchmarks often lack generalization capabilities to accommodate changes in the input domain distribution and, therefore, are inherently limited by the restricted visual and semantic information contained in the original training set. In this thesis, we argue the importance of versatility of deep neural architectures and we explore it from various perspectives. In the first part, we address the ability of deep models to recognize novel semantic concepts without forgetting previously learned ones. First, we define the continual learning problem in semantic segmentation and we propose to retain previous capabilities by distilling knowledge from the previous model at either the feature or output levels of the architecture. Second, we explore how deep model training can be regularized at the latent level - via contrastive learning, matching prototypical representations and sparsity - in order to alleviate catastrophic forgetting of previous concepts while promoting learning of the new ones. Then, we show how we can replace samples of previous categories by using generative networks or web-crawled images. In the second part, we propose novel coarse-to-fine learning tasks where a generic model trained to recognize a coarse set of concepts is progressively updated to recognize finer-grained concepts. In the third part, we investigate deep model generalization to unseen visual domains with no ground truth annotations available. We start with a rigorous definition of the unsupervised domain adaptation task in semantic segmentation and we carefully organize the current literature on the topic. Then, we propose multiple solutions to enable model generalization on a combination of input, feature and output levels. We investigate input-level adaptation via cyclic consistency. Output-level adaptation is enforced by means of adversarial learning schemes relying on a confidence score estimated by the discriminator network. Then, we explore feature-level adaptation via latent regularization enforcing clustering, orthogonality, sparsity and norm alignment among the features. In the last part, we discuss the recent federated learning paradigm in order to train deep architectures in a distributed setting exploiting only data available at decentralized clients and not shared with a central server. We start by defining the general federated learning setup and we analyze its poor robustness to high non-i.i.d. distribution of samples among clients. To mitigate this problem, we propose a naïve federated optimizer which is fair from the users perspective. Then, we introduce a new prototype-guided federated optimizer which has also been evaluated on federated semantic segmentation benchmarks.

Sommario

Il riconoscimento automatico e semantico della scena a livello di pixel è un problema importante in visione computazionale ed un requisito per molte applicazioni pratiche come i veicoli autonomi, i sistemi di navigazione robotica, la videosorveglianza e le applicazioni mediche. Segmentazione semantica è generalmente affrontata con architetture di *deep learning*. Tuttavia, *deep neural networks* su dataset di riferimento pratici spesso mancano di capacità di generalizzazione nell'accomodare cambiamenti nel dominio in ingresso e, pertanto, sono inerentemente limitate dalle ristrette informazioni visive e semantiche contenute nel dataset di allenamento originale. In questa tesi, si sostiene l'importanza della versatilità delle reti neurali e la si esplora da diverse prospettive. Nella prima parte, si affronta l'abilità di *deep models* nel riconoscere nuovi concetti semantici senza dimenticare quelli appresi in precedenza. Dapprima, si definisce il problema di apprendimento continuo in segmentazione semantica e si propone di mantenere le capacità precedenti tramite trasferimento di conoscenza dal modello precedente agendo sul livello latente o su quello dell'output dell'architettura. In secondo luogo, si esplora come l'apprendimento di modelli profondi può essere regolarizzato al livello latente (tramite apprendimento contrastivo, corrispondenza di rappresentazioni prototipiche e sparsità) al fine di alleviare la dimenticanza catastrofica dei concetti precedenti mentre si incentiva l'apprendimento di quelli nuovi. Poi, si mostra come è possibile rimpiazzare i campioni delle categorie precedenti usando reti generative o immagini acquisite dal Web. Nella seconda parte, si propone nuove applicazioni di apprendimento *coarse-to-fine*, dove un generico modello allenato a riconoscere un insieme di concetti grossolani è progressivamente aggiornato per riconoscere concetti più raffinati. Nella terza parte, si indaga la generalizzazione di *deep models* verso domini visivi mai visti senza disponibilità di annotazioni veritiere. Si inizia con una definizione rigorosa dell'adattamento di dominio non supervisionato in segmentazione semantica e si organizza scrupolosamente la corrente letteratura sull'argomento. Poi, si propongono molteplici soluzioni per abilitare la generalizzazione del modello su una combinazione di livelli: ingresso, intermedio e uscita. Si studia l'adattamento al livello di ingresso tramite consistenza ciclica. L'adattamento al livello di uscita è ottenuto tramite schemi di apprendimento avversario basati sul livello di confidenza stimato dalla rete discriminativa. Infine, si esplora l'adattamento sullo spazio latente intermedio tramite regolarizzazione forzando il raggruppamento, l'ortogonalità, la sparsità e l' allineamento delle norme delle features. Nell'ultima parte, si discute il recente paradigma dell'apprendimento federato in modo da allenare *deep models* in modo distribuito sfruttando solamente dati disponibili presso gli utenti decentralizzati e non condivisi con un server centrale. Si inizia definendo lo scenario generale di apprendimento federato e si analizza la sua scarsa robustezza alle distribuzioni di campioni tra i clients fortemente non-*i.i.d.* Per mitigare questo problema, si propone un semplice ottimizzatore federato che è imparziale dal punto di vista degli utenti. Poi si introduce un nuovo ottimizzatore federato guidato dai prototipi, il quale è stato anche valutato su alcuni dataset di riferimento per segmentazione semantica federata.

Contents

Listing of figures

Listing of tables

1 Automatic Semantic Scene Understanding: an Overview

1.1 Introduction

Over the last decade, deep learning techniques have evolved rapidly to address a wide variety of tasks considered extremely challenging beforehand, in particular in the computer vision field, where deep learning has achieved human-like performance in many tasks. When considering scenes, we might be interested in different levels of understanding, depending on the situation and on the usage of the scene inspection. Various visual scene understanding tasks have been depicted in Figure 1.1. In *image classification*, a single label is assigned to the whole image and denotes the pre-dominant object in the scene. In *object detection*, the objects are identified by means of a bounding box and a label is assigned to each box. In *image segmentation*, the scene is clustered into regions corresponding to the various objects and structures but the regions are not labeled. *Semantic segmentation*, instead, is the task of assigning to each pixel in the image a label corresponding to its semantic content. For this reason, it is often referred to as a dense labeling task as opposed to other simpler problems where fewer labels are given as output. Semantic segmentation is a wide research field and a large number of approaches have been proposed to tackle it. In particular, deep learning architectures have recently allowed for attaining considerable improvements. Current state-of-the-art approaches are mostly based on the auto-encoder structure which aim to extract global semantic clues, while retaining input spatial dimensionality, starting from the Fully Convolutional Network (FCN) model [1]. Examples of the most popular and successful approaches are DRN [2], PSPNet [3] and DeepLab [4–6]. Recent reviews on this topic can be found in [7–9].

In the following chapters, we mainly focus on semantic segmentation, as it represents one of the most challenging tasks in automatic visual analysis and allows for a deeper understanding of the image content when compared to simpler problems like image classification or object detection, and it will be a strong prerequisite of many applications (*e.g.*, ranging from robot sensing, to autonomous driving, video surveillance, virtual reality, and many others).

However, when approaching real-world applications different problems arise.

- In some cases, the final target task may not be completely defined at the start of model training with new classes or new tasks being added at run-time. Continual learning strategies deal with this setting aiming at progressively learning the new tasks or classes without

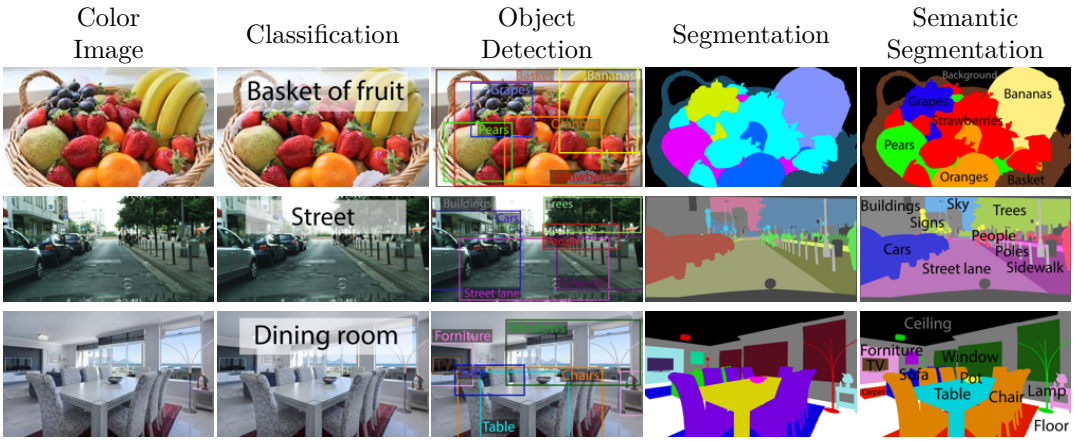


Figure 1.1: Overview of possible visual tasks on a few sample images from classification (sparse learning task) to semantic segmentation (dense learning task).

re-training the machine learning model from scratch [10, 11]. Coarse-to-fine learning, similarly, exploits the previously trained model on a coarse set of classes, to aid the training of the model on a finer set of visual concepts [12].

- In other cases, a large amount of training data for the considered setting and problem is not available. While very large generic datasets are publicly available, the acquisition and labeling of a large amount of data for a specific setting is typically too expensive and time consuming for most companies developing machine learning systems. This gave rise to domain adaptation techniques which allow for transferring the learned knowledge from a generic source dataset to the target data of the problem at hand. These can be either partially supervised, *i.e.*, a small amount of labeled data for the target set is available, or unsupervised, when no labeling information or even no data at all, is available for the target domain [13, 14].

- In some other cases, training data is available only at decentralized clients and it contains sensitive information that cannot be shared with a central server authority for standard supervised model training. In such cases, the domain distribution can be highly different among the different clients and the clients own too few data to train complex model architectures up to convergence. Federated learning handles this need by iteratively computing local solutions and sending them to a central server, which aggregates them and produces a global model ideally able to deal with the different data distributions [15].

This general discussion applies to many learning models and target problems, but becomes extremely relevant when a huge amount of data and a large computational effort for training are needed. In particular, this is the case for dense-predictive image and video understanding problems where each input point is associated to a semantic category.

1.2 Transfer Learning

The problem of knowledge transfer in machine learning was first introduced by Bucilua *et al.* [16]. At that time, the focus was on compressing an entire ensemble of models (a collection of models

whose final predictions are averaged) into a single and simpler model that is easier and faster to train. This concept was further developed in [17] to enable addition of new classes without losing the model's performance on the older set of classes.

In the following, we provide a formal definition of the *transfer learning* problem. Let us define a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is the space of input data and $P(X)$ is the probability distribution function over that input data. A task $\mathcal{T}$ over the domain $\mathcal{D}$ is a combination of a label space $\mathcal{Y}$ with the predictive function $f(\cdot)$ modeling the conditional probability distribution $P(Y|X)$. Thus, any supervised machine learning problem can be generally attributed to the search for a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ which better approximates the unknown underlying $f(\cdot)$, by examining a set of labeled training samples drawn from the joint distribution $P(X, Y)$ over $\mathcal{X} \times \mathcal{Y}$.

In the classical Unsupervised Domain Adaptation (UDA) scenario, the input data domain $\mathcal{D}$ is not unique, *e.g.*, there exist separate source $\mathcal{D}_S$ and target $\mathcal{D}_T$ domains. In the classical Continual Learning (CL) scenario, the input domain $\mathcal{D}$ is split into multiple pieces $\mathcal{D}^{(t)}, t = 1, ..., T_{max}$ available for training at separate times. Furthermore, over these domains different tasks may need to be solved, *e.g.*, two different tasks $\mathcal{T}_S$ and $\mathcal{T}_T$ could have been respectively chosen for the source and target domains or there can be a sequence of tasks $\mathcal{T}^{(t)}, t = 1, ..., T_{max}$ to be solved at different stages t of the learning process. Then, *transfer learning* is defined as seeking for an improved predictive function $f_T(\cdot)$ over the target domain (or over multiple target domains), relying on useful information extracted from source task $\mathcal{T}_S$ on $\mathcal{D}_S$, in case $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

Throughout the thesis, we will appreciate how domain adaptation, continual learning and coarse-to-fine learning can be viewed as special cases of transfer learning: in the first case, the source and target domain are different while the task is the same; in the second case the macro-domain is the same (but it is made available in separate portions) and the task changes; in the third case the domain remains unaltered while the label space changes. Finally, in Federated Learning (FL) each client only experiences its own local distribution and samples are not shared, while the task is to learn a global model to solve all the local tasks: knowledge in this case is transferred through the model parameters periodically sent from the clients to a central server.

1.3 Contributions

This dissertation describes a journey towards pixel-accurate visual understanding in the wild. Namely, we present contributions in four main research areas highly interconnected, which naturally cluster the chapters of this thesis into parts.

1. **Continual Learning (CL).** The first part contains techniques to mitigate the so-called *catastrophic forgetting* phenomenon of deep neural networks when learning new tasks. In this context, we will introduce:

   - the first investigations [18, 19] of continual learning applied to semantic segmentation, without retaining previous images and evaluating on standard semantic segmentation benchmarks;
   - experiments with state-of-the-art and novel knowledge distillation techniques and parameter freezing in [18, 20];
   - in [21] we investigate the latent space organization, proposing new strategies to preserve feature extraction knowledge related to previously seen categories, such as prototype matching, contrastive learning and feature sparsity;

- in [22] we present the first method to use replay data (either from a GAN or from the Web) in continual semantic segmentation, coupling it with a novel background inpainting strategy to generate pseudo-labels to overcome the so-called background shift [23].

2. **Coarse-to-Fine Learning.** The second part explores coarse-to-fine refinements of the label space and finds methods to improve semantic recognition of fine-grained categories with the aid of coarse-level predictions. In this regard, our contributions are:

   - we define and tackle the semantic-level coarse-to-fine learning task aiding the fine-grained network with the output of the coarse-level network [24];
   - we address spatial-level coarse-to-fine learning, where the coarse object-level output is processed by a semantic embedding network and acts as a conditioning while learning part-level categories, in conjunction with a novel adjacency graph-based module that matches the relative spatial relationship between ground truth and predicted parts [25].

3. **Unsupervised Domain Adaptation (UDA).** The third part addresses the well-known domain shift problem, where deep learning architectures are adapted to unseen visual domains without supervision. Our work in this field brings the following contributions:

   - we carefully analyze the existing literature on the topic of UDA in semantic segmentation [13, 19] outlining that adaptation can occur at the input-, feature- or output-levels, and we grouped each strategy by the employed methodology;
   - we investigate output-level adaptation by means of an adversarial learning module that exploits both labeled synthetic data and unlabeled real data, a self-teaching strategy applied to unlabeled data and a region growing framework guided by the segmentation confidence [26, 27];
   - we propose an adaptive confidence mechanism over both: different classes and different training steps [28];
   - we examine the combination of state-of-the-art input-level adaptation based on CycleGAN [29] and a first investigation of feature-level adaptation by means of a couple of discriminator networks in [30];
   - we explore novel feature-level strategies based on feature clustering, orthogonality and sparsity [31];
   - we extend previous latent-level regularization approaches and we couple them with norm alignment [32, 33].

4. **Federated Learning (FL).** The fourth and last part discusses how deep learning architectures for vision tasks can be trained in a decentralized fashion with private data not shared with a central server entity [15]. In particular, we will present:

   - an analysis of the baseline federated learning optimizer (FedAvg) when data is distributed highly non-i.i.d. across clients and a novel federated optimizer, FairAvg, which is unbiased and fair from the user perspective; [34]
   - a new method, FedProto, which computes client deviations using margins of prototypical representations learned on distributed data, and applies them to drive federated optimization via an attention mechanism [35] and we propose three methods to analyze the statistical properties of feature representations learned in FL.

1.4 Outline of the Thesis

The outline of the thesis follows the order of the aforementioned contributions.

Part I presents the continual learning task applied to semantic segmentation. Chapter 2 first introduces the problem and then discusses recent state-of-the-art methods for continual semantic segmentation. In Chapter 3 we explore the effect of parameter freezing and knowledge distillation approaches at various layers of the segmentation network. Chapter 4 presents innovative latent-level regularization methods to reduce forgetting of past knowledge. Chapter 5 investigates the usage of replay data (either coming from a pre-trained GAN, or from the Web) and background inpainting while learning subsequent tasks.

The second part of this thesis (Part II) focuses on the coarse-to-fine refinement of a deep learning architecture initially trained on a coarse set of classes and further refined to solve a finer-grained task. Chapter 6 addresses this problem from two closely-related configurations: Section 6.2 tackles coarse-to-fine learning at the semantic level, where macro (coarse) classes are progressively refined into micro (fine) classes via stacking of multiple architectures. Section 6.3, instead, addresses coarse-to-fine learning at the spatial level, where object-level (coarse) classes are refined into sub-parts (fine) classes via a novel graph matching procedure to match spatial relationship among parts..

Part III presents the unsupervised domain adaptation task in semantic segmentation. Chapter 7 first introduces the problem and then carefully reviews and categorizes the wide literature on the topic. In particular, there are three levels where network adaptation may occur: at the output, feature or input level. Chapter 8 presents our two contributions on output-level UDA. Chapter 9 presents our three contributions on input- and feature-level UDA. First, we explore a combination of a standard input-level adaptation method based on CycleGAN and a simple feature-level method based on a couple of discriminator networks. Second, we explore latent-level adaptation more precisely, aiming at disentangling the latent representations by means of feature clustering, orthogonality and sparsity. Third, we further extend and refine previous considerations on latent-level adaptation methods and we introduce a novel metric to quantify UDA approaches.

The last part of this dissertation (Part IV) is devoted to federated learning and it opens up new horizons towards private and distributed model training. First, Chapter 10 defines the problem in general terms and shows the effect of extreme non-i.i.d. data distribution among clients, introducing a naïve federated optimizer fair from the user perspective. Then, Chapter 11 proposes a novel prototype-guided federated learning optimizer which can also handle federated training of semantic segmentation models.

Finally, Chapter 12 concludes the dissertation summarizing the findings and the open future directions.

Part I

Semantic Recognition of New Categories in the Wild

2

Continual Learning of New Semantic Concepts

This chapter formally introduces and defines the continual learning problem applied to semantic image segmentation [19]. This will serve as a starting point and a reference for the upcoming chapters, where the proposed methodologies are presented and evaluated.

2.1 A Gentle Introduction to Continual Learning

Recent deep learning techniques have rapidly advanced to solve a wide variety of computer vision tasks. Deep learning architectures have matured along the way, however, practical applications soon raised the need for techniques able to improve the learned knowledge over time in order to accomplish new tasks without forgetting previous knowledge. This represents the building paradigm of continual learning in its essence. In other words, when deep learning models are deployed into the real-world, we would like to have the possibility of improving the capability of the models with new experiences or concepts, without retraining them from scratch [19].

In general, the main issue of these computational models is that they are prone to *catastrophic forgetting* [36–39], *i.e.*, training a model with new information interferes with previously learned knowledge and typically greatly degrades the performance. Catastrophic forgetting has been faced even before the rise of neural networks popularity [40–42] and more recently has been rediscovered and tackled in different ways. Deep learning models, in particular, assume that all the data samples are available during the training phase and, therefore, they require that the training is performed on the entire dataset in order to adapt to changes in the data distribution. When trained on sequential tasks with samples progressively available over time, the performance significantly decreases on previously learned tasks as the network parameters are optimized for the new task without accounting for the old ones, if no ad-hoc provisions are employed [10, 43].

Continual learning (also called incremental learning, lifelong learning or never ending learning), then, is the set of techniques designed to face this challenging scenario in which a sequence of tasks comes in succession [44]. Despite being a long-standing problem in computational models [38], in deep learning it has been tackled with some successes only recently.

To further prove its relevance, it is worthwhile to consider an analogy between machine learning and human learning. Indeed, humans encounter a continual stream of learning tasks and are able to generalize to similar unseen tasks [45]. To understand the brain mechanisms and to translate them into computational models, many connectionist and biologically-inspired attempts have been made throughout the years [46, 47]. Recently, the problem has been actively

investigated in some image-level visual tasks (*i.e.*, with one or few labels per each image) such as image classification [48–50] and object detection [51,52]. In dense labeling tasks such as semantic segmentation, the problem has been faced only recently [18, 20, 23, 53–56] due to the inherent increased complexity. Recently, continual learning has been also explored in reinforcement learning [57, 58].

Before digging into the definition of continual learning and exploring its application to dense labeling tasks, we point out some general reviews on the topic, not directly dealing with semantic segmentation. We refer to [59] for catastrophic forgetting in connectionist models, while [10] is the first review to critically compare recent works about the phenomenon in deep learning models. In [60] many approaches are compared into a common framework and in [11] the challenges of continual learning are described with a special focus on robotics.

We start by formally introducing this problem in Section 2.2 and we review the state of the art in sparse learning tasks in Section 2.3. Then, we present the wide range of experimental scenarios that can be considered in Section 2.4, while Section 2.5 presents how these can be tackled by means of knowledge preservation and regularization strategies.

2.2 Problem Statement

Continual Learning (CL) could be regarded as a particular case of transfer learning, where the data domain distribution changes at every incremental step and the model should perform well on all the distributions. It is also strongly connected to the domain adaptation problem that will be the focus of the next part of this thesis, however, in this case, the focus is devoted toward both the input data and the annotations, whose distributions change over time and their number may be increased as well (*i.e.*, more classes to be distinguished).

Due to the intrinsic variety of challenges in continual learning and their respective difficulties, most of the approaches relax the general setup of continual learning to an easier one of incremental task learning. In the latter scenario, tasks are received one at the time and training is performed on the available training data. Hence, this represents a mitigation of the true continual learning system, which is more likely to be encountered in practice [60]. For instance, in class-incremental learning, the learned model is updated to recognize new classes whilst preserving knowledge about previous ones. An overview of this setup is reported in Figure 2.1. In more formal terms, we consider the t-th incremental step (with $t = 1, 2, ..., T_{max}$) and we are given the previous model $\mathcal{M}_{t-1}$ and two sets of data $\left\{\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\right\}$ randomly drawn from the distribution $\mathcal{D}^{(t)}$, which is an observation (or subset) of the complete domain $\mathcal{D}$. Here, $\mathcal{X}^{(t)}$ denotes a set of data samples for step t and $\mathcal{Y}^{(t)}$ denotes the corresponding ground truth annotations (*i.e.*, a single label for the image classification problem or a dense labeling map for the semantic segmentation problem). In the considered class-incremental setup, we assume that each step corresponds to a different learning task.

To mimic what happens in many real-world scenarios and to reduce the need for storage or privacy limitations, most of the frameworks do not store any sample of data $\left\{\mathcal{X}^{(s)}, \mathcal{Y}^{(s)}\right\}$ for any step s preceding the current step t. Hence, the problem becomes even more challenging as the goal is to control an objective function targeting all seen tasks without having access to previous samples. More formally, the empirical risk minimization framework translates into the research of the optimal parameters θ^* by optimizing:

$$\text{argmin}_\theta \sum_{t=0}^{T} \mathbb{E}_{\left(\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\right)} \left[\mathcal{L}\left(\mathcal{M}_t\left(\mathcal{X}^{(t)}; \theta\right), \mathcal{Y}^{(t)}\right)\right] \tag{2.1}$$

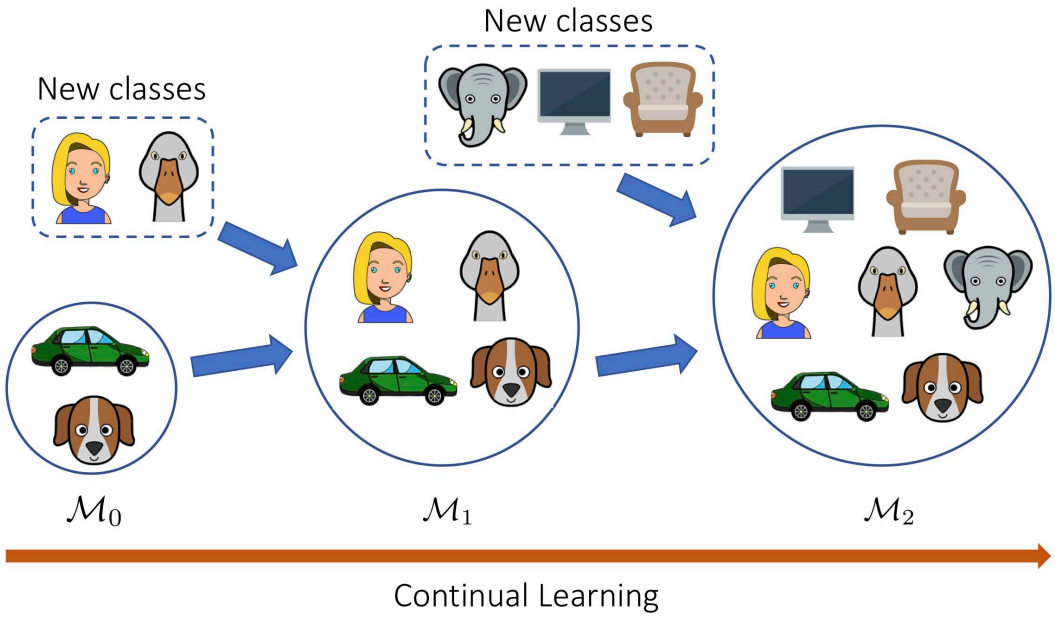


Figure 2.1: Graphical representation of the class-incremental continual learning framework. The model is updated to recognize new classes over time without forgetting previously learned ones.

with model's parameters θ, loss function $\mathcal{L}$, T incremental tasks seen so far, and $\mathcal{M}_t$ the model function at step t. We remark, however, that this objective function cannot be optimized directly as old samples may not be present at all or may be very limited (depending on the continual learning scenario, see Section 2.4). We refer to the case in which all samples are available from the beginning as *joint* (or *offline*) *training*, representing an upper bound of the performance of a continual learning system (*i.e.*, a single stage of training with all samples).

Furthermore, it is useful to gain insight of the problem in terms of marginal output and input distributions, *i.e.*, $P(\mathcal{Y}^{(t)})$ and $P(\mathcal{X}^{(t)})$ respectively, of a generic step t. In general, task-incremental learning considers that $P(\mathcal{Y}^{(t+1)}) \neq P(\mathcal{Y}^{(t)})$ due to $P(\mathcal{X}^{(t+1)}) \neq P(\mathcal{X}^{(t)})$ and that the task output spaces differ over time, *i.e.*, $\{\mathcal{Y}^{(t)}\} \neq \{\mathcal{Y}^{(t+1)}\}$. In the Unsupervised Domain Adaptation (UDA) scenario, instead, we generally observe $P(\mathcal{Y}^{(t+1)}) \neq P(\mathcal{Y}^{(t)})$ due to $P(\mathcal{X}^{(t+1)}) \neq P(\mathcal{X}^{(t)})$, but $\{\mathcal{Y}^{(t)}\} = \{\mathcal{Y}^{(t+1)}\}$, with the number of incremental steps $T_{max} = 1$ and a typically sudden change in the data domain distribution, while changes are in general more gradual in continual learning [60, 61]. We will analyze this more deeply in Chapter 7.

Finally, we remark that ideal continual learning setups consider infinite and continuous stream of training data and at each step the system receives some new samples drawn non-i.i.d. from the current distribution $\mathcal{D}^{(t)}$ that could itself experience sudden or gradual changes with no notification. This is what future approaches should aim to tackle.

2.3 Continual Learning in Sparse Learning Tasks

First Continual Learning (CL) approaches arose in sparse learning tasks, such as image classification or object detection, for the sake of simplicity. We briefly review some of them in this section before introducing continual semantic segmentation techniques in Section 2.5.

Most of the current techniques falls into the following categories [10,60]: dynamic architectures [62–65], regularization approaches [66,67], parameter isolation [68–70] and replay-based methods [71–75].

Early approaches [62,64,65] exploit **dynamic architectures** which grow over time as a tree structure in a hierarchical manner as new classes are observed. Istrate *et al.* [76] propose a method that partitions the original network into sub-networks which are then gradually incorporated in the main one during training. Sarwar *et al.* [77] grow the network incrementally over time while sharing portions of the base module. Dai *et al.* [78] propose a grow-and-prune approach. First, the network grows new connections to accommodate new data; then, the connections are pruned on the basis of the magnitude of weights.

Regularization-based approaches are by far the most widely employed and mainly come in two flavours, *i.e.*, penalty computing and knowledge distillation [17]. Penalty computing approaches [66,79,79] protect important weights inside the models to prevent forgetting. Knowledge distillation [49, 71, 80, 81] is another way of retaining high performance on old tasks which has recently gained wide success. This technique was originally proposed in [17] and [16] to preserve the output of a complex ensemble of networks when adopting a simpler network for more efficient deployment. The idea was adapted to maintain unchanged the responses of the (teacher) network on the old tasks whilst updating a new (student) model with new training samples in different ways. Various approaches have been presented in recent studies [48–51, 71, 82, 83]. Shmelkov *et al.* [51] propose an end-to-end learning framework where the representation and the classifier are learned jointly without storing any of the original training samples. Li *et al.* [49] distill previous knowledge directly from the last trained model. Dhar *et al.* [81] introduce an attention distillation loss as an information preserving penalty for the classifiers' attention maps. In [82] the current model distills knowledge from all previous model snapshots, of which a pruned version is saved. Deep Model Consolidation [84] proposes the idea to train a separate model for the new classes, and then combine the two models (for old and new data, respectively) via double distillation objective. The two models are consolidated via publicly available unlabeled auxiliary data. The Similarity-Preserving Knowledge Distillation (SPKD) [85] strategy aims at preserving the similarities between features of samples of the same class. Recently, novel schemes have been proposed, *e.g.*, to model the information flow through the various layers of the teacher model in order to train a student model to mimic this information flow [86]. Another strategy is to use network distillation to efficiently compute image embeddings with small networks for metric learning [87].

Parameter isolation approaches [70, 88] reserve a subset of weights for a specific task to avoid degradation. Similar strategies consist in freezing or slowing down the learning process in some parts of the network. Kirkpatrick *et al.* [79] developed Elastic Weight Consolidation (EWC) to remember old tasks by slowing down the learning process on the important weights for those tasks. Oquab *et al.* [89] preserve the learned knowledge by freezing the earlier and the mid-level layers of the models. Similar ideas are used in recent studies [49, 76]. Aljundi *et al.* [90] introduce the idea that when learning a new task, changes to important parameters can be penalized, effectively preventing meaningful knowledge related to previous tasks from being overwritten. Shmelkov *et al.* [51] experiment on freezing either all the layers (except for the last) or part of them. Mandziuk *et al.* [91] try to identify and freeze a compact subset of features (nodes) in the hidden layers, that are crucial for the current task, thus preventing forgetting in the future. Similarly, Kirkpatrick *et al.* [79] remember old tasks by slowing down the learning process on the relevant weights for those tasks. Jung *et al.* [92] try to maintain the performance on old tasks by freezing the final layer and discouraging the change of shared weights in feature extraction layers.

Replay-based models exploit either stored [48, 50, 74] or generated [71–73] examples during

the learning process of new tasks. Keeping a small portion of data belonging to previous tasks is another strategy used by some works to preserve the accuracy on old tasks when dealing with new problems [48, 50, 54, 66, 93–95]. In those works the exemplar set to store is chosen according to different criteria. In [48, 93] the authors use an episodic memory which stores a subset of the observed examples from previous tasks, while incrementally learning new classes. Chaudhry *et al.* [94] keep a fraction of samples from previous classes to alleviate intransigence of a model, *i.e.*, the inability of a model to update its knowledge. Hou *et al.* [95] try to balance between preservation and adaptation of the model via distillation and retrospection by caching a small subset of randomly picked data for old tasks. In [50], the classifier and the features used to select the samples for the representative memory are learned jointly in an end-to-end fashion and herding selection [96] is used to pick them. A controlled sampling of memories for replay is proposed by [97], where samples which are most interfered, *i.e.*, whose prediction will be most negatively impacted by the foreseen parameters update, are chosen. Generative Adversarial Networks (GANs) have also been used by some recent methods [71, 73] to generate images containing previous classes instead of storing old classes data, thus retaining high accuracy on old tasks.

2.4 Experimental Setups in Continual Semantic Segmentation

Despite being a quite recent field, continual learning in semantic segmentation already comes in different flavors. In particular, existing works differ in the consideration of the domain distributions $\mathcal{D}^{(t)}$ and of the data sampling $\{\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\}$. The different choices emerge from different target applications. Let us denote with $\mathcal{S}^{(t-1)}$ the previous label set, which is expanded with a set of new classes $\mathcal{C}^{(t)}$ at step t, yielding a new label set $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \cup \mathcal{C}^{(t)}$. As typically assumed in task-incremental settings, the sets of new labels discovered at each step are disjoint, except for a special *background* or *void* class whose behavior and meaning depends on the selected scenario. There are many possible scenarios and one of the key differences lies in the way the background class is considered, which is typical of many semantic segmentation benchmarks. Previous approaches proposed four main different scenarios:

1. **Sequential masked.** This setup reflects the simplest idea on continual semantic segmentation; *i.e.*, each learning step contains a unique set of images, whose pixels belong either to novel classes or to a void class, which is not predicted by the model and it is masked out from both the results and the training procedure. This setup has been used in [54, 55].

2. **Sequential.** This setup has been proposed in [18,20]. Each learning step contains a unique set of images, whose pixels belong to classes seen either in the current or in the previous learning steps. At each step, labels for pixels of both novel classes and old ones are present; however, the specific occurrence of a particular old class is highly correlated to the set of classes being added. For example, if the set of all old classes is $\mathcal{S}^{(t-1)} = \{chair, airplane\}$ and the set of classes being added is $\mathcal{C}^{(t)} = \{dining\ table\}$, then it is reasonable to expect that $\{\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\}$ contains some images with the *chair* class, that typically appears together with the *dining table*, while the class *airplane* is extremely unlikely to occur.

3. **Disjoint.** This setup has been proposed in [20, 23]. At each learning step, the unique set of images is identical to the sequential setup. The difference with respect to the sequential setup lies in the set of labels. At each step, only labels for pixels of novel classes are present, while the old ones are labeled as background in the segmentation maps (this causes the *background* class to change distribution at each step).

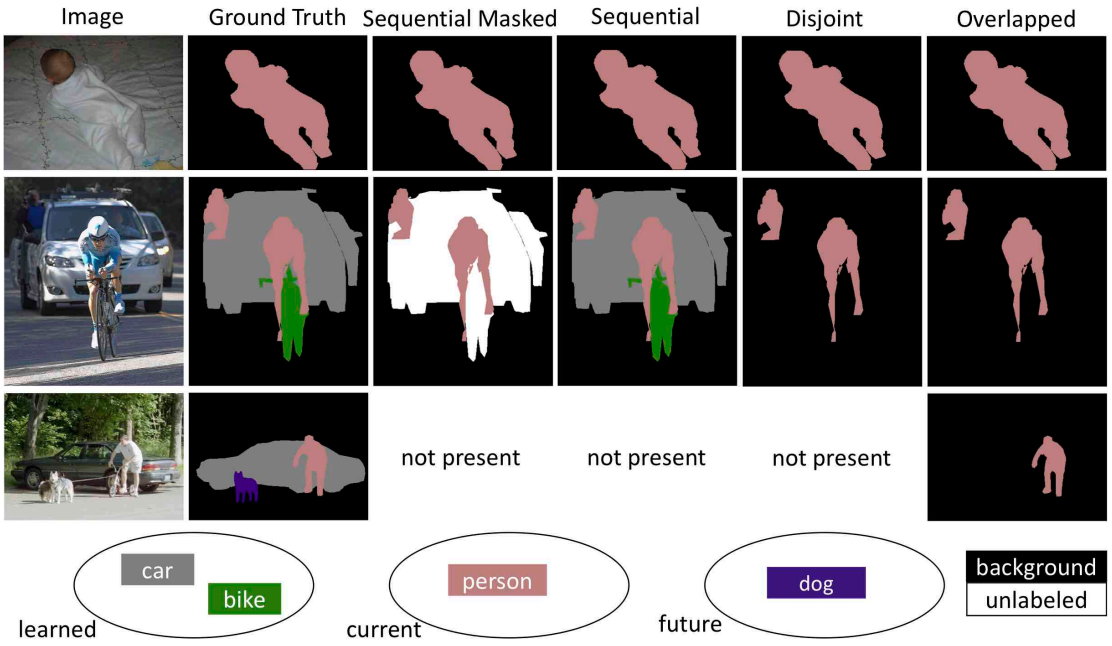


Figure 2.2: Overview of the different setups for class-incremental continual learning in semantic segmentation. The black class represents the $background$ class and the white one represents the *void/unlabeled*.

4. **Overlapped.** This setup moves from the work of [51] for object detection and has been addressed in [21,23,56] for semantic segmentation. In this setup, each training step contains all the images that have at least one pixel of a novel set of classes, with only the classes of the set annotated and the rest set to *background*. Differently from the other settings, in this scenario images may contain pixels of classes that will be learned in the future, but labeled as *background* in the current step; for this reason, as in the previous setting, the *background* class changes distribution at every incremental step.

A few examples of the different semantic map annotations are given in Figure 2.2. Although being subcategories of the same problem they lead to substantially different setups requiring flexible strategies to tackle them.

This complex scenario is getting even more articulated as there exist many different ways of sampling the sets $\mathcal{C}^{(t)}$ of unseen classes and of selecting its cardinality $|\mathcal{C}^{(t)}|$, leading to completely different experiments. Previous work [51, 84, 98] sort the classes using a pre-defined order in the exploited dataset (*e.g.*, alphabetical order). Another possibility is to sort the classes based on their occurrence throughout the dataset [20], to reflect the idea that, in real-world application, it is more likely to start from common classes and introduce rarer ones later. In our works, we experimented both the possibilities. With respect to the second aspect, there are many ways of selecting the cardinality of the sets $\mathcal{C}^t$, leading to different incremental scenarios: one may add a single class, a batch of classes or multiple classes sequentially one after the other [18,20,21,23,56]. All these possibilities open up a very variegate picture, which is being explored only recently, hence many research directions remain still unexplored.

2.5 Techniques for Continual Semantic Segmentation

In this section we are going to review the main methods to tackle task-incremental semantic segmentation grouped by employed technique. We also refer the interested reader to some relevant works in task-incremental image classification, this related field being more explored and mature with respect to continual semantic segmentation.

2.5.1 Knowledge Distillation

The first family of approaches we present is the most commonly employed one thanks to its simplicity and efficacy; *i.e.*, knowledge distillation. This technique was originally proposed by [16] and [17] to preserve the output of a complex ensemble of networks when adopting a simpler network for more efficient deployment. The idea was adapted to maintain unchanged the responses of the network on the old tasks whilst updating it with new training samples typically associated to new tasks. This is typically performed applying a constraint (*e.g.*, a loss function) in order to mimic the responses of the previous model in the current one. Its main effect is to act as a powerful regularization term during the learning process of the current classes, often leading to better performance on both previous and current classes (by preserving the capability of recognizing the former set and avoiding the overestimation of the latter set).

Knowledge distillation has been explored in different setups and it is somehow a prerequisite for successful task incremental learning algorithms. In sparse learning tasks, many algorithms use knowledge distillation in different flavors, as we observed in Section 2.3.

These techniques have been found to be extremely effective and reliable also in dense learning tasks. Ozdemir *et al.* [53] extend the image classification model of [49] to image segmentation simply constructing a knowledge distillation loss as the cross entropy between previous and current model's output probabilities. The authors also devise a strategy to select relevant samples of old data for rehearsal, that improves performance, but violates the assumption used in many scenarios of avoiding previous data storage. Tasar *et al.* [54] apply knowledge distillation via cross entropy between previous and current model's output probabilities for each class, as the model predicts binary segmentation maps for each class separately. This work, however, focuses on a very specific setting related to satellite images and has several limitations when applied to generic semantic segmentation problems. Indeed, it considers the segmentation task as a multi-task learning problem, where a binary classification for each class replaces the multi-class labeling. Additionally, [54] stores some patches chosen according to an importance value determined by a weight assigned to each class and some other patches chosen at random. The capabilities on old classes are preserved by storing a subset of old images. However, for large amount of classes and different applications, the methodology does not scale properly. Moreover, storing previously seen data could represent a serious limitation for applications where privacy issues or limited storage budgets are present.

In [18] we are the first to evaluate on a standard semantic segmentation benchmark and to apply knowledge distillation not only at the output level but also at the intermediate feature space to preserve the geometrical relationships of the extracted features. The work is extended in [20], that introduces and compares many knowledge distillation techniques. In particular, distillation on the output layer is enriched by temperature scaling (*i.e.*, rescaling softmax probabilities by a so-called temperature factor) to consider also the uncertainty of the estimations of previous models. Distillation on intermediate feature level is extended to multiple decoding stages and a scheme inspired by Similarity-Preserving Knowledge Distillation [85] is also proposed. Cermelli *et al.* [23] propose a revisited distillation loss on the output level which accounts for the fact that a previous model could have already seen previous classes labeled as *background* (*i.e.*, in the

overlapped setup). This is coupled with an unbiased weights initialization rule to deal with the atypical behavior of the background class in the *disjoint* and *overlapped* scenarios. The authors initialize the classifier's parameters for the novel classes in such a way that the probability of the background is uniformly spread among the novel classes, preventing the model to be biased toward the background class when dealing with unseen classes. Klingner *et al.* [55] propose a masked and weighted distillation loss on the output level to improve the accuracy on small or under-represented classes within the dataset. Finally, [56, 99] apply a matching distillation to retain both long-range and short-range statistics at different feature levels and at different scales between the old and current model.

Chapter 3 will present our works [18, 20] proposing novel knowledge distillation schemes for continual semantic segmentation.

2.5.2 Parameter Freezing

One of the major achievements of the early connectionist works is that they identified one main strategy to address catastrophic forgetting: *i.e.*, by freezing part of the network weights [48]. This technique has been applied by a large number of contemporary approaches as a regularization attempt to prevent knowledge degradation caused by upcoming tasks.

Following the successes in sparse learning tasks [51, 79, 91, 92] (see Section 2.3), parameter freezing has been proposed as a way to prevent forgetting also in dense labeling tasks. In [18] we propose to freeze all the layers of the encoder in order to preserve unaltered the feature extraction capabilities and only train the decoding parameters. In [20] we propose to freeze only the first couple of layers of the encoder, to preserve the most task-agnostic part of the feature extractor. Cha *et al.* [100] freeze the backbone network and past classifiers to help stability. However, the choice of which layers to freeze remains an open question and an intrinsic trade-off emerges between the capability of efficiently learning new tasks and the preservation of the acquired knowledge. A first attempt of automatic selection of which layers to freeze has been recently introduced by [101] checking the most plastic layers of the network.

2.5.3 Geometrical Feature-Level Constraining

The analysis of the latent space organization is becoming crucial towards understanding and improvement of deep neural networks [102–105]. Recently, some attention has been devoted to latent regularization in continual image classification [106, 107] and in unsupervised domain adaptation [31, 32]. The key idea of these approaches is to disentangle the intermediate feature space in different ways, spacing apart feature representations belonging to different classes. In continual learning, indeed, this can reduce the overlap of the class-wise latent representations when future classes are introduced in the model.

The first paper exploiting this idea in dense tasks is our work [21], where the latent space is constrained to reduce forgetting whilst improving the recognition of novel classes. The framework is driven by three main components: first, prototype matching enforces latent space consistency on old classes, constraining the encoder to produce similar latent representation for previously seen classes in the subsequent steps; second, feature sparsification allows to make room in the latent space to accommodate novel classes; third, contrastive learning is employed to cluster features according to their semantics while tearing apart those of different classes. This work [21] will be the main focus of Chapter 4.

Recently, [56, 99] propose a multi-scale pooling distillation scheme that preserves long- and short-range spatial relationships at the feature level. Furthermore, they design an entropy-

based pseudo-labelling of the background with classes predicted by the old model to deal with background shift and avoid catastrophic forgetting of the old classes.

2.5.4 Replay-based Learning

Rehearsal Learning approaches store some of the past examples and replaying them while learning new information. This problem is exacerbated in continual segmentation where the storage cost is high and images are partially labeled. In [99] a memory-efficient object rehearsal learning procedure is proposed: objects are stored and carefully pasted through selective erasing of foreground objects. In [100] a tiny exemplar set of class-balanced categories is stored into memory to improve both plasticity and stability.
Generative Replay methods train generative models on the current data distribution. Hence, afterwards, it is possible to sample data from past experience when learning on new data. By learning on actual data mixed with artificially generated past data, they try to preserve past knowledge while learning the new task. The generative model is generally a GAN [108] as in [71,73] or an auto-encoder as in [109,110]. Up to date, there are only two approaches for semantic segmentation belonging to this category [22, 111]. In the first one [22], we employ a pre-trained conditioned GAN to generate realistic samples of previous classes which are interleaved during the learning of the new classes. The mapping between old classes of the current dataset and the classes of the pre-trained GAN is performed automatically via a class mapping module translating between the two domains. This large amount of weakly labeled data are assigned to a pseudo-label for semantic segmentation using a side labeling module, requiring only minimal extra storage. Notice that only the weak classification labeling is available for generated data and some pseudo-labels need to be estimated for segmentation. In the second work [111], instead, replay images are synthesized using the image-level labels. Along with real images of new classes, the synthesized images are fed into the distillation-based framework to train the new segmentation model which retains the information about previously learned classes, whilst updating the current model to learn the new ones.
Webly-based Learning models [112,113], *i.e.*, models that learn from samples acquired via web searches, could be an extremely powerful tool to retrieve faithful past examples using as queries the label names of the old classes to preserve. This approach is proposed in [22], where novel samples are interleaved with web-crawled ones (querying the class names to drive the search). This simple method outperforms by a large margin competing approaches with the (minimal) requirement of searching for pictures on the Web. Also in this case, only weak classification labels are available and some pseudo-labeling scheme needs to be introduced.

Chapter 5 will focus on replay-based learning strategies (both generative and web-based) in continual semantic segmentation [22].

2.6 Employed Datasets

In the following chapters we will evaluate the effectiveness and the robustness of the proposed methodologies on a few publicly available datasets that are briefly introduced. Sample scenes taken from the datasets are reported in Figure 2.3.

The **Pascal VOC2012** dataset [114] contains $10,582$ variable-sized images in the training split and $1,449$ in the validation split. The semantic labeling assigns the pixels to 21 different classes (20 plus the background). Since the test set has not been made available, the results are typically reported on the images belonging to the Pascal VOC2012 validation split (*i.e.*, using the validation split as a test set) [51, 84]. A couple of sample scenes are reported as first two

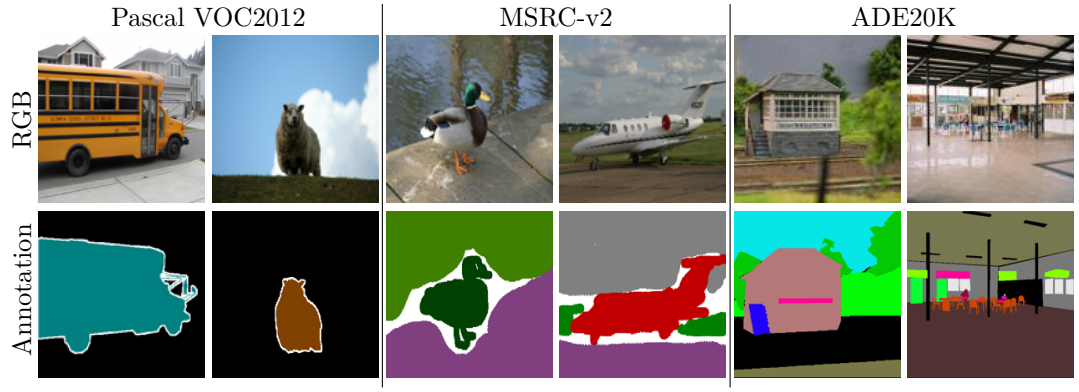


Figure 2.3: Sample scenes from the three employed datasets in the first part of the thesis.

columns of Figure 2.3, where we can appreciate that object-level (foreground) classes are labeled, while other ones are labeled as background.

The **MSRC-v2** dataset [115] consists of 335 images in the *trainval* split and 256 in the test split with variable resolution. It is annotated using 23 semantic classes, however the *horse* and *mountain* classes have been excluded as suggested by the dataset creators [115, 116], because they are underrepresented inside the dataset. The results are computed and reported on the original test set. A couple of sample scenes are reported in columns 3 and 4 of Figure 2.3, where we can see that both foreground and background classes are labeled. However, the annotation quality is coarser and the regions related to the edges of the objects are not labeled.

The **ADE20K** [117] is a large-scale dataset of 22,210 images, 2,000 of which form the validation split. The typical benchmark defined in [117] includes 150 classes, representing both stuff (*e.g.*, sky, building) and object classes (*e.g.*, bottle, chair), differently from VOC2012. A couple of sample scenes are reported as last two columns of Figure 2.3. Both objects and stuff categories are labeled and the ground truth segmentation maps are carefully annotated.

Finally, the accuracy of a CL system is often measured as its ability to mitigate catastrophic forgetting (via accuracy measures). Beyond accuracy, other metrics related to memory, storage and computational efficiency have been proposed [118].

3 Knowledge Distillation from a Teacher Model

3.1 Introduction

Deep learning architectures have shown remarkable results in scene understanding problems, however they exhibit a critical drop of performance when they are required to learn incrementally new tasks without forgetting old ones. This catastrophic forgetting phenomenon impacts on the deployment of artificial intelligence in real-world scenarios where systems need to learn new and different representations over time.

In this chapter, we tackle the catastrophic forgetting problem thoroughly investigating knowledge distillation techniques and we adapt them to the semantic segmentation task [18, 20]. In this way, we retain the information about learned classes, whilst updating the current model to learn the new ones. We develop four main methodologies of knowledge distillation working on both output layers and internal feature representations. Differently from many existing methodologies, we consider the most challenging setting where images from old tasks are not stored and cannot be used to drive the incremental learning process. This is particularly important for the vast majority of real-world applications where old images are not available due to storage requirements or privacy concerns. Hence, only the last model is used to preserve high accuracy on previously learned classes. Specifically, we re-frame the distillation loss concept and we propose four novel approaches where knowledge is distilled from the output layer, from layers of the decoding phase, from intermediate and custom feature layers.

To the best of our knowledge, in [18, 20] we are the first to study incremental learning for semantic segmentation which not retaining previously seen images and evaluating on standard real-world benchmarks.

Extensive experimental results on the Pascal VOC2012 and MSRC-v2 datasets demonstrate that the proposed framework is robust in many different settings and across different datasets, without any previous sample available and even without labeling previous classes in new samples. The proposed schemes allow not only to retain the learned information but also to achieve higher accuracy on new tasks, thanks to the regularizing effect brought by our methodologies, leading to substantial improvements in all the scenarios with respect to the standard approach with no knowledge distillation.

3.1.1 Contributions

The main contributions of our work are the following:

1. we are the first to investigate continual semantic segmentation not retaining previous images and evaluating on standard segmentation benchmarks;

2. we design distillation schemes acting on different levels of the network architecture;

3. we propose to freeze (some layers of) the encoder to preserve unaltered the most task-agnostic part of the feature extraction network;

4. extensive experiments are conducted on many different scenarios. The results are reported on the Pascal VOC2012 and on the MSRC-v2 dataset to validate the generalization properties of the proposed methods.

3.2 Knowledge Distillation for Semantic Segmentation

In this section, we first introduce the class-incremental learning framework, then we present the network architecture employed in our work, finally, we present a set of methodologies based on different types of knowledge distillation strategies [18, 21].

3.2.1 Class-Incremental Continual Learning Framework

As anticipated in Section 2.2, the incremental learning task when referring to semantic segmentation is defined as the ability of a learning system (*e.g.*, a neural network) to learn the segmentation and the labeling of new classes without forgetting (*i.e.*, deteriorating) the performance on previously learned classes. Typically, in semantic segmentation old and new classes coexist in the same image, and the algorithm needs to account for the accuracy on new classes as well as the accuracy on old ones. The first should be as large as possible in order to learn new classes, while the second should be as close as possible to the accuracy experienced before the addition of the new classes, thus avoiding catastrophic forgetting. The critical issue lies in finding the optimal trade-off between preservation of previous knowledge and capability of learning new tasks.
The considered problem is even harder when no data from previous tasks can be preserved, which is the scenario of interest in the majority of the applications where privacy concerns or limited storage requirements subsist. Here the most challenging incremental framework is addressed, in which: previously seen images are not stored nor used; new images may contain examples of unseen classes combined together with pixels belonging to old ones; the approach must scale well with respect to the number of classes.

Let us assume that we are provided with a training dataset $\mathcal{D}^{tr}$. Each pixel of images in $\mathcal{D}^{tr}$ are associated to a unique element of the set $\mathcal{T} = \{c_0, c_1, c_2, ..., c_{C-1}\}$ of C possible classes. In case a background class is present we associate it to the first class c_0 because it has a special and non-conventional behavior being present in almost all the images and having by far the largest occurrence among all the classes.

Moving to the incremental learning steps, we assume that we have trained our network to recognize a subset $\mathcal{S}_0 \subset \mathcal{T}$ of *seen* classes using a labeled subset $\mathcal{D}_0^{tr} \subset \mathcal{D}^{tr}$, whose images contain only pixels belonging to the classes in $\mathcal{S}_0$. We then perform some incremental steps $k = 1, 2, ...$ in which we want to recognize a new subset $\mathcal{U}_k \subset \mathcal{T}$ of *unseen* classes in a new set of training steps. During the k-th incremental step the set of all previously learned classes is denoted as $\mathcal{S}_{k-1}$ and after the current step, the new set $\mathcal{S}_k$ will contain also the last added classes. Formally, $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \mathcal{U}_k$ and $\mathcal{S}_{k-1} \cap \mathcal{U}_k = \emptyset$. Each step of training involves a new set of samples, *i.e.*, $\mathcal{D}_k^{tr} \subset \mathcal{D}^{tr}$, whose images contain only elements belonging to $\mathcal{S}_{k-1} \cup \mathcal{U}_k$. Notice that this set is

disjoint from previously used samples, *i.e.*, $\left(\bigcup_{j=0,\dots,k-1} \mathcal{D}_j^{tr}\right) \cap \mathcal{D}_k^{tr} = \emptyset$. It is important to notice that images in $\mathcal{D}_k^{tr}$ could also contain classes belonging to $\mathcal{S}_{k-1}$, however their occurrence will be limited since $\mathcal{D}_k^{tr}$ is restricted to consider only images which contain pixels from at least one class belonging to $\mathcal{U}_k$. The specific occurrence of a particular class belonging to $\mathcal{S}_{k-1}$ is highly correlated to the set of classes being added (*i.e.*, $\mathcal{U}_k$). For example, if we assume that the set of old classes is $\mathcal{S}_{k-1} = \{car, sofa\}$ and the set of new classes is $\mathcal{U}_k = \{bus\}$, then it is reasonable to expect that $\mathcal{D}_k^{tr}$ contains examples of the class *car*, that appears in road scenes together with the *bus*, while the *sofa* is extremely unlikely to occur. If not stated otherwise, we evaluate our techniques on the sequential case defined in Section 2.4.

In this work we experimented using both an alphabetical ordering of classes and an order based on the occurrence of the classes in the dataset. Additionally, following [51, 84] we deeply analyze the behavior of our algorithms when adding a single class, a batch of classes and when making multiple incremental steps each with one or more classes.

3.2.2 Network Architecture

The methods proposed in this section can be fitted into any deep network architecture; however, since most recent architectures for semantic segmentation are based on the auto-encoder scheme, we focus on this representation. In particular, for the experimental evaluation of the results we use the DeepLab-v2 network [5], with ResNet-101 as the backbone, whose weights were pre-trained [119] on MSCOCO [120] or ImageNet [121]. The pre-training of the feature extractor, as done also in other incremental learning works as [49], is needed since VOC2012 and MSRC-v2 datasets are too small to be used for training a complex network like the DeepLab-v2 from scratch. However, MSCOCO data are used only for the initialization of the feature extractor since the labeling information of this dataset could contain information in some way related to the classes to be learned during the incremental steps. To further evaluate the impact of the pre-training, we also tried a different initialization of the backbone using the ImageNet [121] dataset for image classification (see Section 3.6.1). As previously introduced, the DeepLab-v2 model is based on an auto-encoder structure (*i.e.*, it consists of an encoder part followed by a decoder phase) where the decoder is composed by Atrous Spatial Pyramid Pooling (ASPP) layers in which multiple atrous convolutions with different rates are applied in parallel on the input feature map and then merged together to enhance the accuracy at multiple scales. The original work exploits also a post-processing step based on Conditional Random Fields, but we removed this module to train the network end-to-end and to measure the performance of the incremental approaches without considering the contribution of post-processing steps not related to the training.

3.2.3 Proposed Method

The proposed incremental learning schemes for semantic segmentation are now introduced: a general overview of the proposed approach is shown in Figure 3.1. We start by training the chosen network architecture to recognize the classes in $\mathcal{S}_0$ with the corresponding training data $\mathcal{D}_0^{tr}$. As detailed in Section 3.2.1, $\mathcal{D}_0^{tr}$ contains only images with pixels belonging to classes in $\mathcal{S}_0$. The network is trained in a supervised way with a standard cross-entropy loss. After training, we save the obtained model as M_0.

Then, we perform a set of incremental steps indexed by $k = 1, 2, ...$ to make the model learn every time a new set of classes $\mathcal{U}_k$. At the k-th incremental step, the current training set $\mathcal{D}_k^{tr}$ is built with images that contain at least one of the new classes (but they can possibly contain also pixels belonging to previously seen classes). During step k, the model M_{k-1} is loaded and

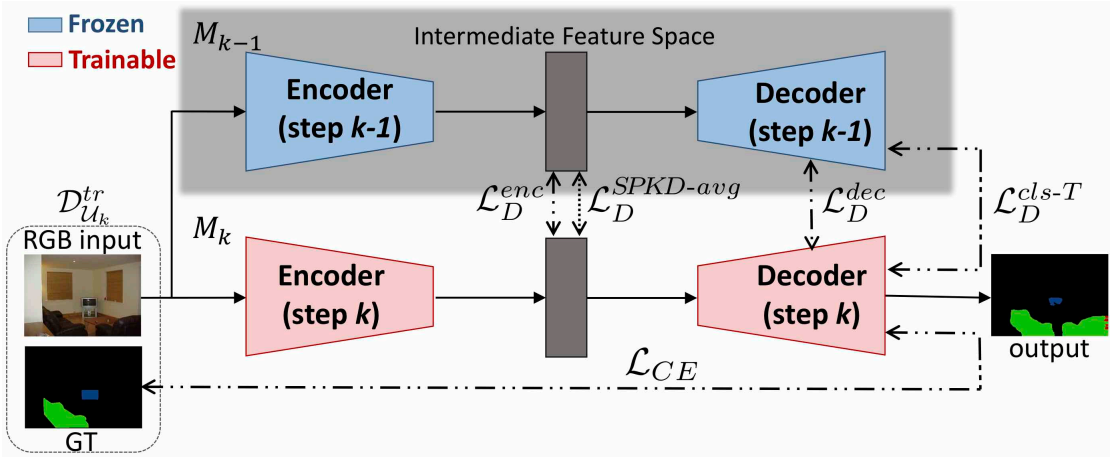


Figure 3.1: Overview of the k-th incremental step of our learning framework for semantic segmentation of RGB images. The scenario in which the current model M_k is completely trainable, *i.e.* not frozen, is reported. The model M_{k-1}, instead, is frozen and is not being updated during the current step.

trained exploiting a linear combination of two losses: a cross-entropy loss $\mathcal{L}_{CE}$, which learns how to identify and label the classes, and a distillation loss $\mathcal{L}_D$, which retains knowledge of previously seen classes and will be detailed in the following. After the k-th incremental step, we save the current model as M_k and we repeat the described procedure every time there is a new set of classes to learn.

The loss $\mathcal{L}$ used to train the model is defined as:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_D \mathcal{L}_D \tag{3.1}$$

where $\mathcal{L}_D \in \{\mathcal{L}'_D, \mathcal{L}''_D, \mathcal{L}'''_D, \mathcal{L}''''_D\}$ is one of the various distillation loss models which will be detailed in the following, while λ_D is an experimentally tuned parameter balancing the two terms. Setting $\lambda_D = 0$ corresponds to the *fine-tuning* scenario in which no distillation is applied and the cross-entropy loss is applied to both unseen and seen classes. We expect this case to exhibit some sort of catastrophic forgetting, as already pointed out in the literature.

During the k-th incremental step, the cross-entropy loss $\mathcal{L}_{CE}$ is applied to all the classes. It is defined as:

$$\mathcal{L}_{CE} = -\frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathbf{X}_n \in \mathcal{D}_k^{tr}} \sum_{c \in \mathcal{S}_k} \mathbf{Y}_n[c] \cdot \log\left(M_k\left(\mathbf{X}_n\right)[c]\right) \tag{3.2}$$

where $\mathbf{Y}_n[c]$ and $M_k\left(\mathbf{X}_n\right)[c]$ are respectively the one-hot encoded ground truth and the output of the segmentation network corresponding to the estimated score for class c. Note that, since $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \mathcal{U}_k$, the sum is computed on both old and newly added classes, but since new ones are much more likely in $\mathcal{D}_k^{tr}$, there is a clear unbalance toward them leading to catastrophic forgetting [122].

As regards the distillation loss $\mathcal{L}_D$, we focus on losses that only depend on the previous model M_{k-1} to avoid the need for large storage requirements.

3.2.4 Distillation on the Output Layer ($\mathcal{L}_D^{cls\text{-}T}$)

The first considered distillation term $\mathcal{L}'_D$ for semantic segmentation is the cross-entropy loss computed on already seen classes between the probabilities produced by the output of the softmax layer of the previous model M_{k-1} and the output of the softmax layer of the current model M_k (if we assume to be at the k-th incremental step). Notice that the cross-entropy is computed only on already seen classes, *i.e.*, on classes in $\mathcal{S}_{k-1}$, since we want to guide the learning process to preserve the behavior on such classes. The distillation loss is then defined as in [18]:

$$\mathcal{L}_D^{cls} = -\frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathbf{X}_n \in \mathcal{D}_k^{tr}} \sum_{c \in \mathcal{S}_{k-1}} M_{k-1}\left(\mathbf{X}_n\right)[c] \cdot \log\left(M_k\left(\mathbf{X}_n\right)[c]\right) \tag{3.3}$$

Furthermore, we improve the model by rescaling the logits using a softmax function with temperature T, *i.e.*:

$$\sigma(z_c) = \frac{\exp(z_c/T)}{\sum_j \exp(z_j/T)} \tag{3.4}$$

where z_c is the logit value corresponding to class c. Hence, denoting with M_k^T the output of the segmentation network for the estimated score of class c after the procedure of Eq. (3.4), we can rewrite Eq. (3.3) as:

$$\mathcal{L}_D^{cls\text{-}T} = -\frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathbf{X}_n \in \mathcal{D}_k^{tr}} \sum_{c \in \mathcal{S}_{k-1}} M_{k-1}^T\left(\mathbf{X}_n\right)[c] \cdot \log\left(M_k^T\left(\mathbf{X}_n\right)[c]\right) \tag{3.5}$$

Intuitively, when $T > 1$ the model produces a softer probability distribution over classes thus helping to retain information about the uncertainty of the classification scores [17, 123]. In the experiments we empirically set T ranging from 1 to 10^3 depending on the scenario. Temperature scaling was not present in the conference version of the work [18] and it reveals to be useful especially when one class is added at a time.

When the new task is quite similar with respect to previous ones, the encoder E, which aims at extracting some intermediate feature representation from the input information, could be frozen to the status it reached after the initial training phase (we call it E_F in short). In this way, the network is constrained to learn new classes only through the decoder, while preserving the features extraction capabilities unchanged from the training performed on $\mathcal{S}_0$. We evaluated this approach both with and without the application of the distillation loss in Eq. (3.3) and Eq. (3.5). Since the procedure of freezing the whole encoder could appear too restrictive (and in fact it does not scale well to the addition of a large number of classes), we tried also to freeze only the first couple of convolutional layers of the encoder (*i.e.*, the first two layers of the ResNet-101 network): we call this version E_{2LF}. Freezing only the first layers allows to preserve the lower level descriptions while updating the weights of the task-specific layers of the encoder and of the decoder. A comparison of the different encoder freezing schemes is shown in Figure 3.2.

3.2.5 Distillation on the Intermediate Feature Space ($\mathcal{L}_D^{enc}$)

Another approach to preserve the feature extraction capabilities of the encoder is to apply a distillation loss on the intermediate level corresponding to the output of the encoder E_k, *i.e.*, on the feature space before the decoding phase. The distillation function working on the feature space in this case can no longer be the cross-entropy but rather a geometrical penalty. At that level, indeed, the considered layer is not anymore a classification layer but instead just an internal stage where the output should be kept close to the previous one in, *e.g.*, Frobenius norm. We

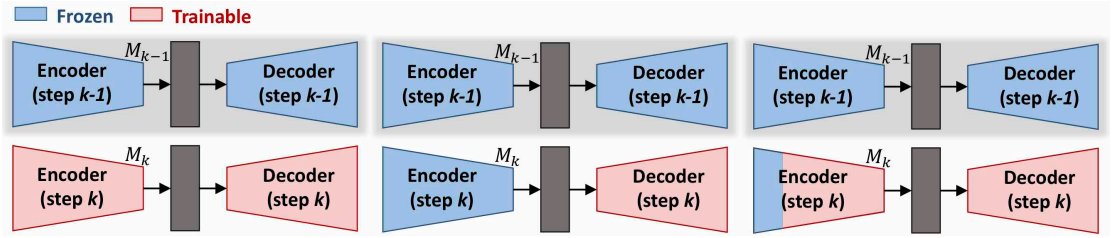


Figure 3.2: Comparison of the different freezing schemes of the encoder at the k-th incremental step. The whole model at previous step, *i.e.* M_{k-1}, is always completely frozen and it is employed only for knowledge distillation purposes.

also considered using L1 loss, but we verified empirically that both L1 and cross-entropy lead to worse results. Considering that the network corresponding to model M_k can be decomposed into an encoder E_k and a decoder, the distillation term becomes:

$$\mathcal{L}_D^{enc} = \frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathbf{X}_n \in \mathcal{D}_k^{tr}} \|E_{k-1}(\mathbf{X}_n) - E_k(\mathbf{X}_n)\|_F^2 \tag{3.6}$$

where $E_k(\mathbf{X}_n)$ denotes the features computed by E_k when a generic image $\mathbf{X}_n \in \mathcal{D}_k^{tr}$ is fed as input.

3.2.6 Distillation on Dilation Layers ($\mathcal{L}_D^{dec}$)

We also tried to apply geometrical penalties at different points inside the network. In particular, we found that a reliable strategy is to apply the distillation on the four dilation layers contained in the ASPP block of the decoder [5]. Hence, the distillation term becomes:

$$\mathcal{L}_D^{dec} = \frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathbf{X}_n \in \mathcal{D}_k^{tr}} \sum_{i=1}^{4} \frac{\|d_{k-1}^i(\mathbf{X}_n) - d_k^i(\mathbf{X}_n)\|_F^2}{4} \tag{3.7}$$

where $d_k^i(\mathbf{X}_n)$ is the output of the dilation layer d_k^i with $i = 1, 2, 3, 4$ when $\mathbf{X}_n \in \mathcal{D}_k^{tr}$ is fed as input. This strategy was not considered in [18] and proved to be effective in preserving the learned knowledge.

An ablation study on multi-layer knowledge distillation is presented in Section 3.6.3. We can appreciate how distilling early layers of the decoding stage pushes the results toward distillation of the intermediate features (*i.e.*, close to $\mathcal{L}_D^{enc}$), while distilling later layers of the decoding stage pushes the results toward distillation on the output layer (*i.e.*, close to $\mathcal{L}_D^{cls\text{-}T}$).

3.2.7 Similarity-Preserving Distillation on the Intermediate Feature Space ($\mathcal{L}_D^{SPKD}$)

Finally, we introduce a modified version of the Similarity-Preserving Knowledge Distillation (SPKD) [85] aiming at preserving the similarities between features of samples of the same class. Let us denote with $\mathcal{B}$ a training batch containing B images, with W and H the (reduced) spatial dimension in the features' space and with F the number of features channels. In the original version of the SPKD approach, the content of the feature layers $E_k(\mathcal{B}) \in \mathbb{R}^{B \times H \times W \times F}$ is reshaped as $E'_k(\mathcal{B}) \in \mathbb{R}^{B \times HWF}$ and then the matrix $\tilde{A'}_k = E'_k(\mathcal{B}) \cdot E'_k(\mathcal{B})^T \in \mathbb{R}^{B \times B}$ is computed

and row-wise normalized to A'_k. The SPKD loss [85] is then computed as:

$$\mathcal{L}_D^{SPKD} = \frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathcal{B} \in \mathcal{D}_k^{tr}} \frac{1}{B} ||A'_k - A'_{k-1}||_F^2 \tag{3.8}$$

The approach was originally introduced for image classification and is based on the idea that each image contains mostly one object in foreground and is associated to a single label. In practice, it does not capture that, in semantic segmentation, multiple classes co-exist in the same image and that an object belonging to a certain class can be a small part of the image.

For this reason, we introduce a variation of this approach. We accumulate the activations over all spatial locations in the feature space, *i.e.*, we compute the matrix $E''_k(\mathcal{B}) \in \mathbb{R}^{B \times F}$ where $E''_k[b, f] = \sum_{h=1}^{H} \sum_{w=1}^{W} E_k[b, h, w, f]$. The loss is then computed as in the previous case but using matrix E''_k in place of E'_k, *i.e.*, $\tilde{A}''_k = E''_k(\mathcal{B}) \cdot E''_k(\mathcal{B})^T$ and row-wise normalized to A''_k. The loss is then computed as:

$$\mathcal{L}_D^{SPKD\text{-}avg} = \frac{1}{|\mathcal{D}_k^{tr}|} \sum_{\mathcal{B} \in \mathcal{D}_k^{tr}} \frac{1}{B} ||A''_k - A''_{k-1}||_F^2 \tag{3.9}$$

This allows to avoid the dependency on the spatial locations of the objects and reduces the computation time due to the smaller matrices size. We verify the validity of this modification with respect to Eq. (3.8) in Section 3.4.

A summary of the proposed strategies for the incremental learning steps is shown in Figure 3.1, which points out the four losses. As a final remark, we also tried a combination of the described distillation losses without achieving significant enhancements.

3.3 Training Procedure

Datasets: to evaluate the effectiveness and the robustness of the proposed methodologies we choose to employ two publicly available datasets for semantic segmentation: namely, the Pascal VOC2012 [114] and the MSRC-v2 [115] datasets (see Section 2.6). These benchmarks have been widely used to evaluate semantic segmentation schemes [7, 124].

For both datasets, we randomly flipped and scaled the images of a random factor between 0.5 and 1.5 with bilinear interpolation. For training, random crops of 321×321 pixels have been used for memory limitations. The testing phase has been conducted at the original resolution of the images.

Implementation Details: the proposed incremental learning strategies are independent of the backbone architecture and generalize well to different scenarios where new tasks should be learned over time. For the experimental evaluation we select the architecture presented in Section 3.2.2. We optimize the network weights with Stochastic Gradient Descent (SGD) as done in [5]. The initial stage of network training on the set $\mathcal{S}_0$ is performed by setting the starting learning rate to 10^{-4} and training for $|\mathcal{S}_0| \cdot 1000$ steps decreasing the learning rate to 10^{-6} with a polynomial decay rule with power 0.9 [5]. Notice that the number of training steps is linearly proportional to the number of classes in $\mathcal{S}_0$. We employ weight decay regularization of 10^{-4} and a batch size of 4 images.

The incremental training steps $k = 1, 2, ...$ are performed employing a lower learning rate to better preserve previous weights. In this case the learning rate starts from $5 \cdot 10^{-5}$ and decreases to 10^{-6} after $|\mathcal{U}_k| \cdot 1000$ steps of polynomial decay. As before, we train the network for a number of steps which is proportional to the number of classes contained in the considered incremental

Table 3.1: Per-class IoU of the proposed approaches on VOC2012 when the last class, *i.e.*, the *tv/monitor* class, is added.

$M_1(20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	**mIoU old**	tv	**mIoU**	**mPA**	**mCA**
Fine-tuning	90.2	80.8	33.3	83.1	53.7	68.2	84.6	78.0	83.2	32.1	73.4	52.6	76.6	72.7	68.8	79.8	43.8	76.5	46.5	68.4	67.3	20.1	65.1	90.7	76.5
E_F	92.7	86.2	32.6	82.9	61.7	74.6	92.9	83.1	87.7	27.4	79.4	59.0	79.4	76.9	77.2	81.2	49.6	80.8	49.3	83.4	71.9	43.3	70.5	93.2	81.4
E_{2LF}	92.5	85.7	32.6	81.5	60.4	74.3	93.0	83.3	87.5	26.9	79.5	59.2	78.8	76.2	77.5	81.0	49.4	80.4	49.8	83.2	71.6	45.3	70.4	93.2	81.1
$\mathcal{L}_D^{cls}$ [18]	92.0	83.9	37.0	84.0	58.8	70.9	90.9	82.5	86.1	32.1	72.5	51.0	79.9	72.3	77.3	80.9	45.1	78.1	45.7	79.9	70.0	35.3	68.4	92.5	79.5
$\mathcal{L}_D^{cls\text{-}T}$	92.6	85.7	33.4	85.3	63.1	74.0	92.6	83.0	86.4	30.4	78.1	55.0	79.1	77.8	76.4	81.7	49.7	80.2	48.5	80.4	71.7	44.4	70.4	93.2	80.1
E_F, $\mathcal{L}_D^{cls\text{-}T}$	93.1	85.9	37.3	85.5	63.1	77.5	93.2	82.2	88.8	29.4	80.1	57.1	80.6	79.4	76.9	82.5	50.0	81.8	51.1	85.0	**73.0**	**51.9**	**72.0**	**93.6**	82.3
E_{2LF}, $\mathcal{L}_D^{cls\text{-}T}$	92.7	84.7	35.3	86.0	60.7	73.3	92.8	82.6	87.6	29.9	78.6	54.4	80.3	78.0	76.3	81.5	50.0	80.9	49.5	82.8	71.9	47.4	70.7	93.2	80.7
$\mathcal{L}_D^{enc}$	92.9	84.8	36.4	82.6	63.5	75.0	92.2	83.6	88.3	29.5	80.3	59.6	79.7	80.2	78.9	81.2	49.7	78.9	51.0	84.1	72.6	50.6	71.6	93.4	**83.4**
$\mathcal{L}_D^{dec}$	92.2	85.4	34.3	82.4	61.6	73.4	91.7	82.7	86.4	32.4	77.2	57.4	76.3	72.6	76.1	81.1	53.7	79.2	46.1	81.5	71.2	35.6	69.5	92.6	81.5
E_F, $\mathcal{L}_D^{dec}$	92.5	84.7	33.8	80.4	60.8	76.1	91.5	82.9	87.1	29.5	78.4	58.7	76.1	73.7	78.8	81.0	51.1	78.3	48.3	84.9	71.4	42.7	70.1	93.0	82.6
$\mathcal{L}_D^{SPKD}$ [85]	92.5	83.5	35.8	84.3	60.1	71.7	88.9	83.2	87.0	32.0	79.9	57.5	78.7	78.1	77.8	81.0	50.4	80.1	49.5	78.4	71.5	43.0	70.2	92.9	82.0
$\mathcal{L}_D^{SPKD\text{-}avg}$	92.6	84.8	34.8	84.8	61.4	71.9	90.5	83.8	87.4	32.0	80.0	58.1	78.6	77.9	77.6	81.3	50.4	80.6	49.6	80.5	71.9	44.5	70.6	93.1	82.3
$M_0(0-19)$	93.4	85.5	37.1	86.2	62.2	77.9	93.4	83.5	89.3	32.6	80.7	57.3	81.5	81.2	77.7	83.0	51.5	81.6	48.2	85.0	73.4	-	73.4	93.9	84.3
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	73.7	70.5	73.6	93.9	84.2

step thus allowing to automatically adapt the training length to the number of new classes being learned. The considered metrics are the most widely used for semantic segmentation problems: namely, per-class Pixel Accuracy (PA), per-class Intersection over Union (IoU), mean PA (mPA), mean Class Accuracy (mCA) and mean IoU (mIoU) [124].

We use TensorFlow [125] to develop and train the network: the overall training of the considered architecture takes around 5 hours on a NVIDIA 2080 Ti GPU. The code is available online at `https://lttm.dei.unipd.it/paper_data/KDSemantic`.

3.4 Experimental Results on Pascal VOC2012

Following the experimental scenarios presented in [51] and [18], we start by analyzing the incremental learning task with the addition of a single class, in alphabetical or frequency-based order, and then move to the addition of 5 and 10 classes, either all together or sequentially. We firstly present the results on the Pascal VOC2012 dataset and then move to the MSRC-v2. Then, we evaluate the proposed approach in the disjoint setting. Finally, we summarize the main achievements of each proposed strategy.

3.4.1 Addition of One Class

We start from the addition of the last class, in alphabetical order, to our classifier. Specifically, we consider $\mathcal{S}_0 = \{c_0, c_1, ..., c_{19}\}$ and $\mathcal{U}_1 = \{c_{20}\} = \{tv \backslash monitor\}$. The evaluation on the VOC2012 validation split is reported in Table 3.1. The table reports the IoU for each single class and the average values of the pixel and class accuracy. The network is firstly optimized on the train split containing samples belonging to any of the classes in $\mathcal{S}_0$, *i.e.*, $\mathcal{D}_0^{tr}$: we indicate with $M_0(0-19)$ the initial training of the network on $\mathcal{D}_0^{tr}$. The network is then updated exploiting the dataset $\mathcal{D}_1^{tr}$ and the resulting model is referred to as $M_1(20)$. In this way, we always specify both the index of the training step and the indexes of the classes added in the considered step.
From the first row of Table 3.1 we can appreciate that adapting the network in the standard way, *i.e.*, without additional provisions, leads to an evident degradation of the performance with a final mIoU of 65.1% compared to 73.6% of the reference model $M_0(0-20)$, where all the 21 classes are learned at once. This is a confirmation of the catastrophic forgetting phenomenon

in semantic segmentation, even with the addition of just one single class. Furthermore, simple strategies like adding a bias or a scaling factor to the logits of the old classes do not allow to solve this issue. The main issue of the naïve approach (we call it *fine-tuning* in the tables) is that it tends to predict too frequently the last class, even when it is not present, as proved by the fact that the model has a very high pixel accuracy for the *tv/monitor* class of 84.3% but a very poor IoU of 20.1% on the same class. This is due to the high number of false positive detections of the considered class which are not taken into account by the pixel accuracy measure. For this reason semantic segmentation frameworks are commonly ranked using the mIoU score. On the same class, the proposed methods are all able to outperform the naïve approach in terms of IoU by a large margin: the best method achieves a mIoU of 51.9% on the *tv/monitor* class.

Knowledge distillation strategies and the procedure of freezing (part of) the encoder provide better results because they act as regularization constraints. Interestingly, those procedures allow to achieve higher accuracy not only on previously learned classes but also on newly added ones, which might be unexpected if we do not consider the regularization behavior of those terms. Hence, all the proposed strategies allow to alleviate forgetting and all of them overcome the standard approach (without knowledge distillation) in any of the considered metrics, as it can be verified in Table 3.1. We can appreciate that $\mathcal{L}_D'^{cls\text{-}T}$ alone is able to improve the average mIoU by 5.3% with respect to the standard case. Notice how the improved version of $\mathcal{L}_D^{cls\text{-}T}$ with temperature scaling introduced in this work achieves a significant improvement of 2% of mIoU with respect to the conference version of the work [18], that corresponds to $\mathcal{L}_D^{cls}$. Furthermore, it leads to a much better IoU on the new class, greatly reducing the aforementioned false positives issue. If we completely freeze the encoder E without applying knowledge distillation the model improves the mIoU by 5.4%. If we combine the two mentioned approaches, *i.e.*, we freeze E and we apply $\mathcal{L}_D^{cls\text{-}T}$, the mIoU reaches 72.0%, with an overall improvement of 6.9%, higher than each of the two methods alone (also the performance on the new class is higher). If we just freeze the first two layers of the encoder and we apply knowledge distillation, *i.e.*, $M_1(20)[E_{2LF}, \mathcal{L}_D^{cls\text{-}T}]$ (we use the square brackets to collect the list of employed strategies), a slightly lower mIoU of 70.7% is achieved. Instead, if we apply an L2 loss at the intermediate features space ($\mathcal{L}_D^{enc}$) the model achieves 71.6% of mIoU, which is 6.5% higher than the standard approach. It is noticeable that two completely different approaches to preserve knowledge, namely $M_1(20)[E_F, \mathcal{L}_D^{cls\text{-}T}]$ (which applies a cross-entropy between the outputs with encoder frozen) and $M_1(20)[\mathcal{L}_D^{enc}]$ (which applies an L2-loss between features spaces), achieve similar and high results both on new and old classes.

If we apply an L2 loss on the dilation filters of the decoder, *i.e.*, the $\mathcal{L}_D^{dec}$ loss, we obtain a mIoU of 69.5% which is higher than the standard approach but lower than the other strategies. Freezing the encoder yields in this setting to a small improvement from 69.5% to 70.1%. Finally, the original SPKD loss achieves 70.2% of mIoU, while the improved version presented in this work ($M_1(20)[\mathcal{L}_D^{SPKD\text{-}avg}]$) achieves a mIoU of 70.6%.

From the class-wise results we can appreciate that changes in performance on previously seen classes are correlated with the class being added. Some classes have even higher results in terms of mIoU because their prediction has been reinforced through the new training set. For example, objects of the classes *sofa* or *dining table* are typically present in scenes containing a *tv/monitor*, hence in the considered scenario they achieve almost always higher accuracy. Other classes, instead, get more easily lost because they represent uncorrelated objects not present inside the new set of samples. For example, instances of *bird* or *horse* are not present in indoor scenes typically associated with the *tv/monitor* class being added.

Some visual examples are shown in the first two rows of Figure 3.3. We can visually appreciate that knowledge distillation and encoder freezing help in preserving previous classes (*e.g.*, the hand of the *person* in the first row and the *table* in the second are better preserved). At the

Figure 3.3: Qualitative results on sample scenes for the addition of one class. In the first two rows the *tv/monitor* class is added, in the last row the *bottle* class is added.

Table 3.2: Per-class IoU of the proposed approaches on VOC2012 when the last class according to the occurrence in the dataset, *i.e.* the *bottle* class, is added.

$M_1(20)$	backgr.	person	cat	dog	car	train	chair	bus	sofa	mbike	din. table	aero	horse	bird	bike	tv	boat	plant	sheep	cow	**mIoU old**	bottle	**mIoU**	**mPA**	**mCA**
Fine-tuning	91.9	80.7	82.2	72.3	81.7	77.9	27.2	90.2	46.9	74.5	56.1	82.4	71.8	77.9	34.9	55.8	58.7	31.0	71.9	66.9	66.6	63.8	66.5	92.4	75.8
E_F	92.6	81.8	87.8	81.5	83.5	84.1	26.4	92.3	50.6	68.5	54.6	86.1	79.3	85.9	36.6	66.6	62.3	49.6	79.2	80.0	71.5	61.9	71.0	93.3	81.4
E_{2LF}	92.1	82.0	85.2	77.8	82.9	80.5	23.4	90.8	48.7	75.6	54.0	81.6	75.0	77.2	34.7	60.3	57.6	32.6	75.6	70.1	67.9	64.7	67.7	92.7	76.9
$\mathcal{L}_D^{cls\text{-}T}$	92.9	82.7	87.9	80.0	82.3	82.5	31.7	90.5	49.3	75.7	57.0	85.2	77.9	85.5	37.3	65.2	63.7	48.3	79.3	77.4	71.6	68.2	71.5	93.4	81.2
E_F, $\mathcal{L}_D^{cls\text{-}T}$	92.9	82.1	89.3	82.2	83.5	85.0	28.6	92.5	50.2	74.2	55.4	86.1	79.2	85.4	36.9	66.7	62.6	52.1	80.1	79.6	**72.2**	64.2	**71.8**	**93.6**	81.0
E_{2LF}, $\mathcal{L}_D^{cls\text{-}T}$	92.9	82.6	88.2	81.3	82.4	85.3	31.4	91.5	50.1	76.0	57.0	84.8	78.0	85.7	36.9	64.9	61.8	49.3	79.9	76.8	71.8	**69.0**	71.7	93.5	**81.8**
$\mathcal{L}_D^{enc}$	92.9	81.7	88.5	81.8	83.8	85.0	27.2	92.4	51.8	73.0	56.0	85.9	79.9	85.7	37.0	65.7	61.7	48.7	80.1	80.0	71.9	62.3	71.5	93.5	**81.8**
$\mathcal{L}_D^{dec}$	92.5	82.7	86.5	79.7	83.4	83.1	28.4	91.9	46.6	68.7	54.7	83.3	75.4	83.8	32.8	65.8	62.8	48.2	78.6	73.8	70.1	67.5	70.0	93.1	78.9
$\mathcal{L}_D^{SPKD\text{-}avg}$	92.7	82.0	86.8	79.3	83.1	82.5	30.4	91.7	48.3	74.6	55.7	84.8	77.3	84.8	36.0	66.8	62.2	49.0	78.5	76.4	71.1	67.6	71.0	93.3	81.0
$M_0(0-19)$	93.5	80.9	89.7	82.8	84.4	85.5	33.1	92.5	47.6	79.3	57.0	85.9	79.9	85.9	37.2	67.8	62.5	53.4	80.5	79.7	73.0	-	73.0	94.0	83.3
$M_0(0-20)$	93.4	83.6	89.7	82.4	82.4	86.4	35.4	92.7	51.1	76.8	52.9	85.4	82.0	85.7	36.7	70.5	63.3	52.3	82.4	80.9	73.3	78.7	73.6	93.9	84.2

same time it does not compromise the learning of the new class (*e.g.*, the *tv/monitor* in the first row is quite well-localized). In the second row, however, a challenging example is reported where none of the proposed methodologies, and neither the baseline approach, are able to accurately detect the new class.

In Table 3.2 the IoU results in the same scenario are shown, but this time ordering the classes according to the pixels' occurrence of each class, thus the *bottle* class is added last. Similarly to the previous case the baseline approach exhibits a large drop in performance while knowledge distillation always helps in every scenario. As before, the best performing strategy is $M_1(20)[E_F, \mathcal{L}_D^{cls\text{-}T}]$. A visual example is shown in the last row of Figure 3.3: we can verify that knowledge distillation and encoder freezing help not only to retain previously seen classes (*e.g.*, the *cat* in the example), but also to better detect and localize the new class, *i.e.* the *bottle*, thus acting as a regularization term.

Table 3.3: Per-class IoU of the proposed approaches on VOC2012 when 5 classes are added at once.

$M_1(16-20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	**mIoU old**	plant	sheep	sofa	train	tv	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	89.7	59.5	34.6	68.2	58.1	58.8	59.2	79.2	80.2	30.0	12.7	51.0	72.5	61.7	74.4	79.4	60.6	36.4	32.4	27.2	55.2	42.4	38.7	55.4	88.4	70.6
E_F	90.2	72.4	32.4	57.3	50.5	68.0	10.5	81.5	84.7	25.4	10.6	57.4	76.2	68.6	77.7	79.8	58.9	36.2	30.7	30.3	43.4	54.5	39.0	54.2	88.5	68.2
E_{2LF}	89.6	59.2	35.2	67.8	56.2	58.0	59.3	80.9	81.6	31.3	13.4	54.9	73.3	63.3	73.0	80.0	61.1	37.5	31.9	25.8	56.2	41.4	38.5	55.7	88.4	70.4
$\mathcal{L}_D^{cls\text{-}T}$	91.4	85.0	35.6	84.8	61.8	70.5	85.6	77.9	83.7	30.7	72.3	45.3	76.2	76.9	77.0	81.3	**71.0**	33.8	55.2	30.9	73.9	51.6	**49.1**	**65.8**	**91.6**	**78.1**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	91.7	83.4	35.6	78.7	60.9	73.0	65.8	82.2	87.0	30.2	58.0	55.3	80.0	78.3	78.5	81.4	70.0	36.0	45.9	32.2	62.5	53.0	45.9	64.3	91.5	76.1
E_{2LF}, $\mathcal{L}_D^{cls\text{-}T}$	91.0	80.3	35.8	82.9	60.9	66.4	80.9	80.1	84.3	32.8	59.4	47.7	75.9	76.0	76.4	81.6	69.5	37.7	47.2	29.9	69.8	48.0	46.5	64.0	91.0	77.1
$\mathcal{L}_D^{enc}$	90.9	81.4	33.9	80.3	61.9	67.4	73.1	81.8	84.8	31.3	0.4	55.8	76.1	72.2	77.7	81.2	65.6	39.4	31.8	31.3	64.1	52.9	43.9	60.5	90.0	74.9
$\mathcal{L}_D^{dec}$	91.1	85.1	31.7	80.3	62.6	72.1	82.6	79.5	84.4	31.1	34.9	56.6	77.2	75.7	77.5	81.7	69.0	40.6	43.4	30.3	70.7	52.2	47.4	63.9	91.0	77.4
$\mathcal{L}_D^{SPKD\text{-}avg}$	90.6	81.7	33.0	80.4	61.9	65.4	68.4	80.9	85.6	31.1	4.8	55.1	76.3	63.8	77.3	80.0	64.8	38.5	31.4	29.9	63.2	50.8	42.7	59.5	89.7	74.0
$M_0(0-15)$	94.0	83.5	36.1	85.5	61.0	77.7	94.1	82.8	90.0	40.0	82.8	54.9	83.4	81.2	78.3	83.2	75.5	-	-	-	-	-	-	75.5	94.6	86.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	75.1	52.3	82.4	51.1	86.4	70.5	68.5	73.6	93.9	84.2

3.4.2 Addition of Multiple Classes

In this section we consider a more challenging scenario where the initial training is followed by one incremental step to learn multiple classes.

First, the addition of the last 5 classes at once (referred to as $15-20$) is discussed and the results are shown in Table 3.3. Results are much lower than in the previous cases since there is a larger amount of information to be learned. The baseline exhibits an even larger drop in accuracy because it tends to overestimate the presence of the new classes as shown by the IoU scores of the newly added classes which are often lower by a large margin (see Table 3.3), while, on the other side, the pixel accuracy of the new classes is much higher. In this case, E_F and E_{2LF} fail to accommodate the substantial changes in the input distribution. In this case the distillation on the output layer, *i.e.*, $M_1(16-20)[\mathcal{L}_D^{cls\text{-}T}]$, achieves the highest accuracy. Here, the approaches based on $\mathcal{L}_D^{cls\text{-}T}$ outperform the other ones (even on new classes). Also in this scenario, all the proposals outperform the standard approach on both old and new classes. Interestingly, some previously seen classes exhibit a clear catastrophic forgetting phenomenon because the updated models mislead them with visually similar classes belonging to the set of new classes. For example, the *cow* and *chair* classes are often misled (low IoU and low PA for these classes) with the newly added classes *sheep* and *sofa* that have similar shapes (low IoU but high PA for them).
Qualitative results are shown in Figure 3.4: we can appreciate that the naïve approach tends to overestimate the presence of the new classes in spite of previously learned ones or in spite of the background. This can be seen from the figure, where instances of *train*, *bus* and *sofa* classes (which are added during the incremental step) are erroneously predicted in place of the new class or in the *background* region. These classes are correctly removed or strongly reduced applying the proposed strategies even if there is not a clear winner. On the *aeroplane* in the first row all the proposed approaches work well. Applying $\mathcal{L}_D^{cls\text{-}T}$ and freezing the encoder removes the false detection of the *sofa* in row 2 that is challenging for the other approaches. The *car* in the last row is better detected using the $\mathcal{L}_D^{SPKD\text{-}avg}$ loss.

The next experiment regards the addition of 10 classes at once and the results are shown in Table 3.4. Here knowledge distillation is less effective. Indeed, even though it enhances the results, the improvement is smaller with respect to that in other scenarios. In particular, the idea of freezing only the first two layers of the encoder, introduced in this version of the work, together with knowldege distillation leads to the best results in this setting. The gap is reduced because the fine-tuning approach already achieves quite high results preventing other methods

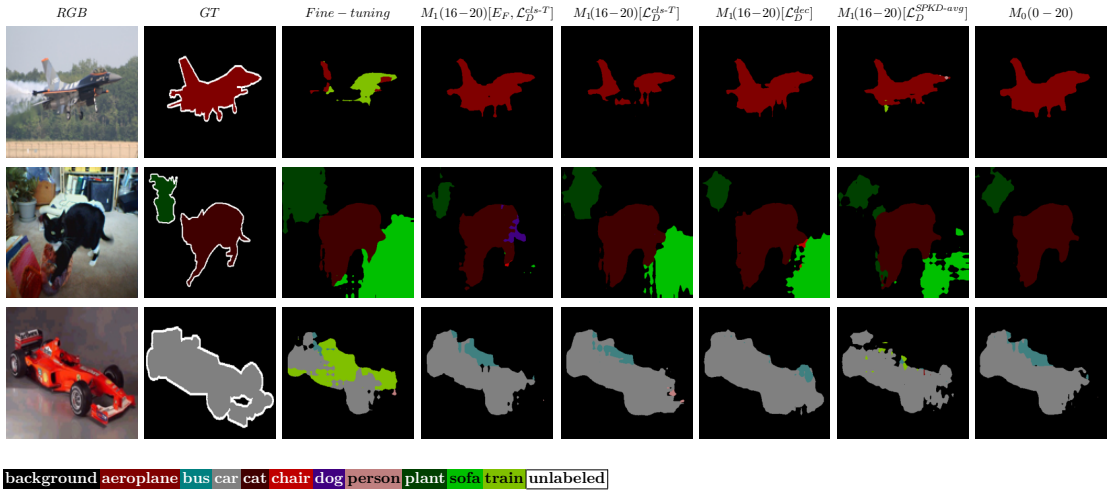


Figure 3.4: Qualitative results on sample scenes for the addition of 5 classes at once. The set of new classes is *plant*, *sheep*, *sofa*, *train* and *tv*

Table 3.4: Per-class IoU of the proposed approaches on VOC2012 when 10 classes are added at once.

$M_1(11-20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	**mIoU old**	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	91.9	82.4	32.0	70.8	61.7	67.7	91.1	79.8	72.7	30.5	61.6	67.5	49.1	70.6	63.4	72.9	79.4	43.5	72.0	44.8	79.1	60.7	**63.5**	65.6	**91.9**	78.2
E_F	91.3	85.0	33.1	82.0	63.2	75.2	89.5	76.5	74.9	25.4	67.7	69.4	42.2	64.4	66.2	68.2	67.7	38.6	69.8	32.9	72.1	59.5	58.2	64.1	91.1	75.0
E_{2LF}	91.8	82.7	32.6	72.9	60.2	67.2	91.3	80.7	75.7	30.7	61.6	68.0	46.9	70.2	65.2	72.2	78.8	43.4	70.1	44.4	81.4	58.6	63.1	65.6	91.8	78.6
$\mathcal{L}_D^{cls\text{-}T}$	91.7	83.2	33.4	80.9	62.3	72.6	89.2	76.8	77.6	28.0	64.1	69.1	48.6	73.5	65.7	72.9	76.6	41.3	74.2	39.5	79.0	62.1	63.3	66.3	**91.9**	77.3
E_F, $\mathcal{L}_D^{cls\text{-}T}$	91.4	85.2	33.3	82.5	62.7	75.1	89.7	76.4	75.3	25.9	67.9	69.6	42.2	64.7	66.4	68.0	67.9	39.4	70.4	32.9	72.5	60.5	58.5	64.3	91.2	75.2
E_{2LF}, $\mathcal{L}_D^{cls\text{-}T}$	91.8	84.0	33.6	83.2	62.7	72.4	90.9	77.0	79.9	28.2	65.4	**69.9**	46.8	72.7	66.8	71.5	75.3	41.1	74.2	38.2	80.0	59.7	62.6	**66.5**	**91.9**	77.7
$\mathcal{L}_D^{enc}$	92.1	83.5	34.0	79.5	61.7	69.1	90.9	78.5	72.5	29.3	61.2	68.4	46.2	66.1	65.3	74.3	79.1	43.0	70.0	47.1	78.3	63.5	63.3	66.0	91.8	**79.4**
$\mathcal{L}_D^{dec}$	92.0	84.5	33.5	74.7	61.2	71.5	89.7	77.9	73.5	28.6	61.8	68.1	51.9	67.1	64.9	70.8	77.3	42.8	70.5	45.3	78.9	61.6	63.1	65.7	**91.9**	77.3
$\mathcal{L}_D^{SPKD\text{-}avg}$	91.9	82.1	32.3	76.4	61.4	66.1	91.3	78.5	72.9	30.0	60.7	67.6	47.1	68.6	65.2	74.8	79.7	42.1	68.9	47.4	78.4	62.2	63.4	65.6	91.8	79.0
$M_0(0-10)$	95.3	86.4	34.4	85.6	69.7	79.3	94.6	87.6	93.1	44.2	91.9	78.4	-	-	-	-	-	-	-	-	-	-	-	78.4	96.1	90.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	74.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	70.5	72.1	73.6	93.9	84.2

to largely overcome it. We argue that the critical aspect is that the cardinality of the set of classes being added is comparable to that of the set of previously learned classes.

3.4.3 Sequential Addition of Multiple Classes

The last set of experiments on VOC2012 regards the addition of one or more classes more than once (*i.e.*, new classes are progressively added instead of all in one shot).

Let us start from the case in which two sets of 5 classes are added in two incremental steps after an initial training stage with 10 classes, leading to the final model $M_2(11-15, 16-20)$. The mIoU results are reported in Table 3.5 where we can appreciate a more severe drop in performance if compared to the introduction of all the 10 classes in a single shot. In particular, the standard approach without distillation leads to a very poor mIoU of 50.2%. Catastrophic forgetting is largely mitigated by knowledge distillation, that in this case proved to be very effective. In the best settings, that in this case are the distillation $\mathcal{L}_D^{cls\text{-}T}$ with E_F and the newly introduced distillation applied to the dilation layers ($\mathcal{L}_D^{dec}$), the mIoU improves of 9.2% with

Table 3.5: Per-class IoU of the proposed approaches on VOC2012 when 5 classes are added two times.

$M_2(1\text{-}15, 16\text{-}20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	**mIoU old**	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	89.3	56.8	32.0	60.6	56.0	55.8	36.8	75.7	76.4	28.2	4.1	52.0	47.3	67.8	50.8	69.2	78.3	34.8	27.6	25.7	44.8	37.0	48.3	50.2	86.9	66.2
E_F	88.5	71.5	29.7	49.9	41.2	68.4	4.4	79.0	80.3	24.8	6.2	49.5	44.7	66.2	38.5	68.1	73.7	36.4	25.3	27.1	46.3	52.7	47.9	48.7	86.5	62.7
$\mathcal{L}_D^{cls\text{-}T}$	90.3	80.7	33.5	74.9	62.6	62.3	74.2	77.2	78.5	27.4	23.6	62.3	44.3	70.6	61.2	72.5	78.9	38.1	37.5	29.7	62.4	46.9	54.2	58.4	89.5	73.0
E_F, $\mathcal{L}_D^{cls\text{-}T}$	90.3	85.0	32.6	73.2	61.3	72.5	79.2	79.7	81.0	27.1	31.3	**64.8**	42.5	68.5	58.5	68.5	73.1	36.5	33.9	30.5	68.0	54.8	53.5	**59.4**	**89.8**	71.4
$\mathcal{L}_D^{enc}$	89.8	64.7	33.3	73.7	58.3	63.8	48.7	77.9	79.8	28.4	11.4	57.2	50.1	68.2	53.0	70.8	79.2	39.0	28.9	26.8	49.4	44.2	51.0	54.3	88.0	69.7
$\mathcal{L}_D^{dec}$	90.6	81.8	32.9	77.7	62.5	66.7	78.8	78.7	79.2	27.7	25.1	63.8	49.7	69.1	56.6	72.1	79.5	40.1	34.2	28.5	65.5	50.7	**54.6**	**59.4**	89.6	**73.9**
$\mathcal{L}_D^{SPKD\text{-}avg}$	89.8	78.1	30.3	58.6	52.6	65.6	53.8	78.0	74.3	29.9	4.7	56.0	46.9	62.6	49.3	68.6	78.4	32.7	23.0	30.0	61.7	49.0	50.2	53.2	88.0	67.7
$M_0(0-10)$	95.3	86.4	34.4	85.6	69.7	79.3	94.6	87.6	93.1	44.2	91.9	78.4	-	-	-	-	-	-	-	-	-	-	-	78.4	96.1	90.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	74.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	70.5	72.1	73.6	93.9	84.2

Table 3.6: Per-class IoU of the proposed approaches on VOC2012 when 5 classes are added two times with classes ordered based on the occurrence in the dataset.

$M_2(1\text{-}15, 16\text{-}20)$	backgr.	person	cat	dog	car	train	chair	bus	sofa	mbike	din. table	**mIoU old**	aero	horse	bird	bike	tv	boat	plant	sheep	cow	bottle	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	91.3	80.5	75.7	67.8	80.2	73.4	29.7	84.4	42.1	70.4	55.6	68.3	19.7	41.1	5.7	29.8	63.9	41.2	38.4	45.7	55.1	63.3	40.4	55.0	90.3	69.1
E_F	92.4	82	81.5	75.5	82.8	82.9	29.1	92.9	46	71.4	54.9	71.9	70.2	24.5	58	31.7	59.9	46.5	39	49.3	53.1	60.5	49.3	61.1	91.7	72.3
$\mathcal{L}_D^{cls\text{-}T}$	92.5	81.0	78.6	69.9	80.5	80.3	31.0	88.8	45.1	73.3	50.2	70.1	79.7	53.4	71.5	32.7	61.6	53.1	40.1	58.8	59.9	71.0	**58.2**	**64.4**	**92.2**	**76.0**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	92.5	81.7	82.1	76.1	83.5	83.1	29.0	92.7	46.7	71.2	55.4	**72.2**	70.9	29.5	59.2	32.0	59.6	46.3	38.5	49.0	52.4	61.6	49.9	61.6	91.9	72.6
$\mathcal{L}_D^{enc}$	92.5	82.2	82.7	74.2	81.8	78.7	31.8	88.0	46.2	73.8	58.3	71.8	66.0	39.8	56.9	31.0	63.5	42.6	45.3	54.5	60.2	69.3	52.9	62.8	92.0	74.7
$\mathcal{L}_D^{dec}$	92.0	81.2	82.6	68.2	78.3	81.4	29.1	91.3	45.1	71.6	56.9	70.7	0.3	23.4	0.1	23.6	61.6	46.8	44.1	49.6	59.4	70.0	37.9	55.1	91.0	66.8
$\mathcal{L}_D^{SPKD\text{-}avg}$	92.1	81.2	82.9	74.5	82.0	79.5	30.3	88.5	42.3	74.7	55.5	71.2	62.7	25.1	42.9	31.2	61.3	45.4	43.3	47.0	55.9	70.9	48.6	60.4	91.4	73.5
$M_0(0-10)$	92.5	80.6	89.2	85.5	86.3	86.8	30.7	93.3	46.2	80.7	59.6	75.6	-	-	-	-	-	-	-	-	-	-	-	75.6	93.5	82.8
$M_0(0-20)$	93.4	83.6	89.7	82.4	82.4	86.4	35.4	92.7	51.1	76.8	52.9	75.2	85.4	82.0	85.7	36.7	70.5	63.3	52.3	82.4	80.9	78.7	71.8	73.6	93.9	84.2

respect to the standard approach. The method using $\mathcal{L}_D^{dec}$ is also the one obtaining the best mIoU on the new classes.

In Table 3.6 the same scenario is evaluated when classes are sorted on the basis of their occurrence inside the dataset. Also in this case a large gain can be obtained with knowledge distillation which leads to 9.4% of improvement in the best case with respect to fine tuning. We can notice that the old classes are better preserved in this case being also the most frequent inside the dataset. Additionally, some methods struggle in learning new classes needing more samples to detect them.

Then, we move to consider the sequential addition of the last 5 classes one by one, *i.e.*, model $M_5(16 \rightarrow 20)$. The results are reported in Table 3.7 where we can appreciate a huge gain of 20.4% of mIoU between the best proposed method (*i.e.* $M_5(16 \rightarrow 20)[E_F, \mathcal{L}_D^{cls\text{-}T}]$) and the baseline approach. In this case, freezing the encoder and distilling the knowledge is found to be very reliable because the addition of one single class should not alter too much the responses of the whole network. Distilling the knowledge from the previous model when the encoder is fixed guides the decoder to modify only the responses for the new class: in this way the best result is obtained. The evolution of the models' mean performance during the various steps is reported in Table 3.8 where we can appreciate how the drop of performance is distributed during the different steps. In particular, we can notice how the accuracy drop is affected by the specific class being added. As expected, the larger drop is experienced when the classes *sheep* or *train* are added because such classes typically appear alone or with the *person* class, *i.e.*, they are only sparsely correlated with a few other classes. The opposite is true when the classes being added

Table 3.7: Per-class IoU of the proposed approaches on VOC2012 when 5 classes are added sequentially.

$M_5(16 \rightarrow 20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	**mIoU old**	plant	sheep	sofa	train	tv	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	87.9	25.6	29.0	51.2	1.7	57.8	10.5	64.8	80.5	30.8	22.9	52.7	66.8	52.1	51.9	78.1	47.8	**36.5**	44.7	31.8	35.1	17.1	33.0	44.2	86.1	55.7
E_F	90.4	62.6	30.9	81.7	53.9	70.8	57.8	80.7	86.2	27.0	71.5	56.8	75.5	77.2	73.2	78.2	67.1	36.3	65.1	30.1	52.9	34.6	43.8	61.6	90.4	72.2
$\mathcal{L}_D^{cls\text{-}T}$	89.8	51.2	29.9	77.2	15.6	62.0	29.2	78.5	75.7	24.4	55.6	44.8	76.2	62.5	65.6	80.1	57.4	27.0	35.2	30.6	42.3	39.7	35.0	52.3	88.6	63.2
E_F, $\mathcal{L}_D^{cls\text{-}T}$	91.1	73.9	31.9	81.4	59.5	71.9	73.1	82.1	87.1	27.2	77.4	56.4	79.1	79.9	76.1	80.7	**70.5**	31.8	55.8	30.1	**62.3**	**41.4**	44.3	**64.6**	**91.3**	**75.2**
$\mathcal{L}_D^{enc}$	90.3	54.2	28.2	78.4	52.5	69.8	59.5	78.5	86.3	28.8	72.3	57.4	76.3	77.1	65.8	79.3	65.9	36.3	**65.5**	31.6	54.7	38.9	**45.4**	61.0	90.4	71.0
$\mathcal{L}_D^{dec}$	90.2	69.1	31.0	78.4	32.1	61.8	41.9	73.7	83.7	30.0	54.8	52.5	69.5	62.8	61.2	81.0	60.8	30.0	46.5	**32.5**	43.5	30.0	36.5	55.1	89.2	66.5
$\mathcal{L}_D^{SPKD\text{-}avg}$	89.9	70.9	31.3	73.5	43.1	68.3	67.9	77.0	82.2	31.3	22.7	55.2	74.9	57.4	62.3	79.2	61.7	34.2	36.5	31.5	62.0	33.0	39.5	56.4	89.3	69.3
$M_0(0-15)$	94.0	83.5	36.1	85.5	61.0	77.7	94.1	82.8	90.0	40.0	82.8	54.9	83.4	81.2	78.3	83.2	75.5	-	-	-	-	-	-	75.5	94.6	86.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	75.1	52.3	82.4	51.1	86.4	70.5	68.5	73.6	93.9	84.2

Table 3.8: mIoU, mPA and mCA of the proposed approaches on VOC2012 when 5 classes are added sequentially.

	Fine-tuning			E_F			$\mathcal{L}_D^{cls\text{-}T}$			E_F, $\mathcal{L}_D^{cls\text{-}T}$			$\mathcal{L}_D^{dec}$			$\mathcal{L}_D^{enc}$			$\mathcal{L}_D^{SPKD\text{-}avg}$		
	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA
$M_1(16)$	71.2	93.7	82.5	71.8	93.7	84.4	72.4	**94.2**	83.0	72.5	94.1	83.5	**72.9**	**94.2**	**84.5**	72.2	93.9	84.3	71.3	93.7	82.6
$M_2(17)$	53.8	90.0	61.8	59.1	91.2	69.0	68.1	**93.4**	78.5	**68.4**	93.3	**79.5**	68.0	**93.4**	78.6	60.0	91.6	69.4	56.5	90.7	65.6
$M_3(18)$	57.7	87.7	68.7	65.7	90.4	78.0	63.3	90.8	74.5	**66.5**	**91.5**	**79.4**	64.6	90.2	76.9	65.5	90.7	76.8	58.3	88.3	70.1
$M_4(19)$	39.3	85.9	47.4	52.6	89.0	61.4	54.1	89.2	64.3	**61.3**	**90.6**	**72.5**	57.9	89.7	69.0	52.1	89.0	60.6	54.3	89.5	64.9
$M_5(20)$	44.2	86.1	55.7	61.6	90.4	72.2	52.3	88.6	63.2	**64.6**	**91.3**	**75.2**	55.1	89.2	66.5	61.0	90.4	71.0	56.4	89.3	69.3

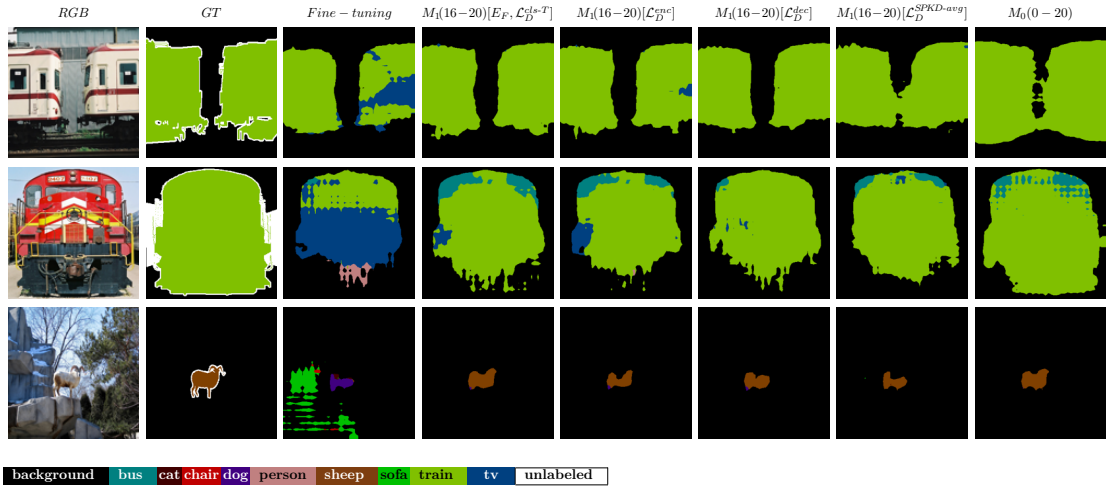


Figure 3.5: Sample qualitative results for the addition of 5 classes sequentially. The added classes are *boat*, *plant*, *sheep*, *cow* and *bottle*.

show high assortativity coefficient with other classes, for example the presence of the classes *potted plant* and *tv/monitor* is highly correlated with the presence of classes like *dining table*, *person* or *chair*. Some visual results for this scenario are reported in Figure 3.5 where a large gap in performance between the naïve approach and some of the best performing proposals can be appreciated. In particular, the standard approach without knowledge distillation tends to overestimate the presence of the last seen class, *i.e.*, *tv/monitor*, in spite of the *background* or of other previously learned classes.

Finally, in Table 3.9 we can appreciate the per-class IoU of the best method of Table 3.7 (*i.e.*,

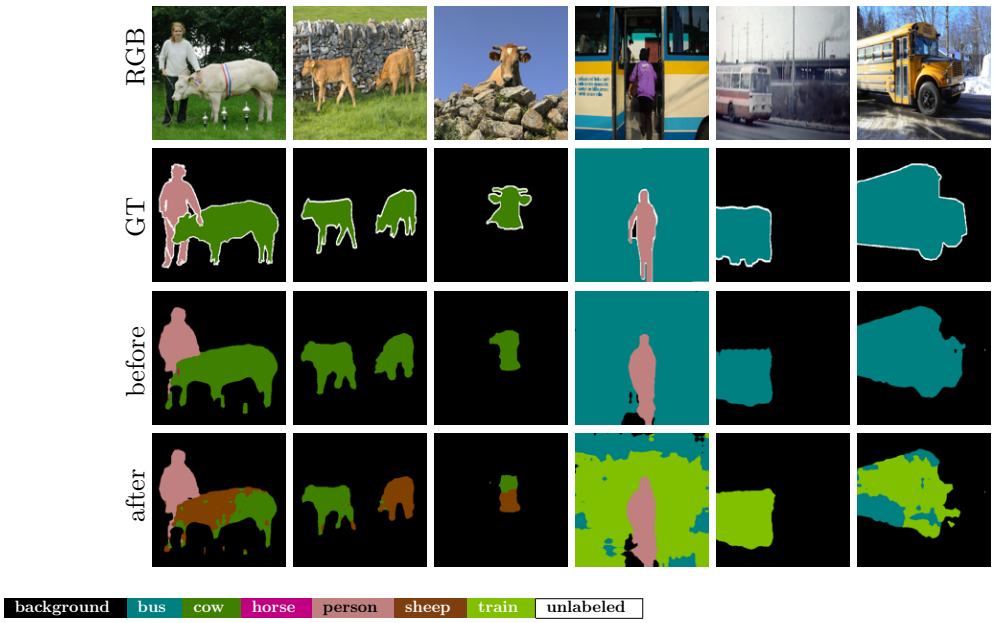


Figure 3.6: Qualitative comparison on sample scenes of the best model of Table 3.9 before and after the addition of a highly correlated class. The first three columns show the performance results after the addition of the *sheep* class while the last three deals with the addition of the *train* class.

Table 3.9: Per-class IoU on VOC2012 when 5 classes are added sequentially. Only the best method of Table 3.7 ("E_F and $\mathcal{L}_D^{cls\text{-}T}$") is reported.

	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mIoU**	**mPA**	**mCA**
$M_0(0-15)$	94.0	83.5	36.1	85.5	61.0	77.7	94.1	82.8	90.0	40.0	82.8	54.9	83.4	81.2	78.3	83.2	-	-	-	-	-	75.5	94.6	86.4
$M_1(16)$	93.5	84.0	36.1	84.8	60.5	72.5	93.4	84.2	89.7	40.0	83.0	55.7	81.9	81.6	79.4	83.2	29.0	-	-	-	-	72.5	94.1	83.5
$M_2(17)$	93.5	84.9	35.6	72.5	61.2	73.7	93.7	83.7	79.6	39.9	73.2	57.1	78.4	74.7	79.1	83.2	29.4	37.3	-	-	-	68.4	93.3	79.5
$M_3(18)$	91.3	83.5	34.4	76.2	61.7	72.6	93.8	83.9	85.6	26.2	77.3	57.4	78.0	77.8	78.8	81.8	30.0	46.7	26.7	-	-	66.5	91.5	79.4
$M_4(19)$	91.2	67.8	31.7	63.9	60.5	73.1	43.2	83.5	86.4	25.1	77.7	56.7	79.1	77.9	74.3	81.7	27.0	49.2	28.0	48.7	-	61.3	90.6	72.5
$M_5(20)$	91.1	73.9	31.9	81.4	59.5	71.9	73.1	82.1	87.1	27.2	77.4	56.4	79.1	79.9	76.1	80.7	31.8	55.8	30.1	62.3	41.4	64.6	91.3	75.2
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	70.5	73.6	93.9	84.2

the method combining $\mathcal{L}_D^{cls\text{-}T}$ and E_F) at each step. An interesting aspect regards the addition of the *sofa* class which causes a tremendous drop of about one third in terms of IoU for the *chair* class given their large visual similarity (from 39.9% to 26.2%). An analogue drop for the same reason is experienced also by the *bus* and *aeroplane* classes when the *train* class is added. Again, when the *sheep* class is added visually similar classes of other animals lose accuracy. These scenarios are depicted in Figure 3.6 where the first three columns show the results of the best model reported in Table 3.9 before and after the addition of the class *sheep* and the last three columns show the results before and after the addition of the *train* class. We can appreciate that regions of the *cow* class which were correctly identified before the addition of *sheep* are then confused with the new class and the same happens with the *bus* and *train* classes.

The same scenario is then analyzed for classes ordered on the basis of their occurrence inside the dataset in Table 3.10. Similar considerations as before hold, however the accuracy on new

Table 3.10: Per-class IoU of the proposed approaches on VOC2012 when 5 classes are added sequentially with classes ordered based on their occurrence.

$M_5(16 \rightarrow 20)$	backgr.	person	cat	dog	car	train	chair	bus	sofa	mbike	din. table	aero	horse	bird	bike	tv	**mIoU old**	boat	plant	sheep	cow	bottle	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	91.2	78.8	81.6	69.5	80.6	69.6	28.6	85.6	41.4	74.9	57.2	77.4	38.5	60.0	34.6	64.3	64.6	27.4	18.3	1.6	46.9	49.7	28.8	56.1	90.7	66.0
E_F	90.2	76.5	84.4	72.9	79.3	70.4	27.4	87.3	44.2	71.5	45.2	75.8	51.6	65.5	32.0	62.4	64.8	51.0	23.9	2.5	56.3	62.1	39.1	58.7	90.6	67.9
$\mathcal{L}_D^{cls\text{-}T}$	92.1	81.4	69.6	66.6	81.5	81.8	26.9	87.5	37.0	71.9	47.0	79.1	46.2	69.0	34.8	65.6	64.9	40.1	**35.3**	11.0	37.0	**66.1**	37.9	58.5	90.9	56.8
E_F, $\mathcal{L}_D^{cls\text{-}T}$	91.1	77.1	84.2	72.9	79.2	75.9	28.0	88.2	43.1	73.4	46.7	78.9	56.6	67.4	32.2	62.9	66.1	**49.8**	29.6	**24.4**	51.3	63.1	**43.6**	60.8	91.1	**70.3**
$\mathcal{L}_D^{enc}$	91.6	82.0	88.8	80.2	83.8	76.8	28.8	91.9	50.0	74.0	54.9	80.6	66.0	72.6	36.1	69.9	**70.5**	34.1	18.4	4.8	**53.0**	56.9	33.4	**61.7**	**92.0**	70.2
$\mathcal{L}_D^{dec}$	92.0	81.7	84.2	72.5	82.0	70.1	32.9	87.6	45.9	72.7	54.0	74.4	61.2	76.2	34.1	69.8	68.2	34.0	24.5	6.5	45.6	60.7	34.2	60.1	91.6	69.3
$\mathcal{L}_D^{SPKD\text{-}avg}$	91.7	81.9	84.5	76.0	81.5	70.5	31.0	89.1	44.6	76.6	56.3	78.2	50.5	69.0	35.9	69.4	67.9	40.1	30.7	11.4	48.8	61.2	38.4	60.9	91.6	**70.3**
$M_0(0-15)$	93.5	81.1	89.3	84.3	84.6	85.4	30.0	92.9	47.5	79.0	57.8	86.0	85.5	84.7	36.4	71.3	74.3	-	-	-	-	-	-	74.3	94.1	84.2
$M_0(0-20)$	93.4	83.6	89.7	82.4	82.4	86.4	35.4	92.7	51.1	76.8	52.9	85.4	82.0	85.7	36.7	70.5	75.1	63.3	52.3	82.4	80.9	78.7	68.5	73.6	93.9	84.2

Table 3.11: Per-class IoU of the proposed approaches on VOC2012 when 10 classes are added sequentially.

$M_{10}(11 \rightarrow 20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	**mIoU old**	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	86.7	29.0	28.0	49.7	2.2	54.4	1.5	54.7	75.3	29.4	7.8	38.0	46.5	60.7	17.0	23.3	75.1	29.1	38.0	31.5	27.6	11.6	36.0	37.1	83.6	49.0
E_F	87.9	65.2	28.6	73.1	56.8	70.0	73.4	77.5	80.1	26.9	54.5	63.1	46.8	56.4	44.1	40.2	74.5	36.8	35.6	27.4	58.7	28.4	44.9	54.4	85.8	59.4
$\mathcal{L}_D^{cls\text{-}T}$	87.0	28.1	27.4	47.4	0.2	48.6	1.6	59.3	73.1	22.9	27.9	38.5	42.2	56.9	21.8	30.9	77.1	28.4	27.7	24.5	33.1	21.6	36.4	37.5	83.8	51.1
E_F, $\mathcal{L}_D^{cls\text{-}T}$	89.5	66.1	28.3	72.7	58.3	70.7	74.0	78.2	80.3	27.7	55.1	**63.7**	45.7	56.2	**45.6**	41.5	74.8	37.2	36.9	26.7	59.2	**28.6**	45.2	54.9	88.5	**67.1**
$\mathcal{L}_D^{enc}$	89.1	54.7	28.7	75.3	44.5	69.4	73.2	79.4	83.4	30.2	54.9	62.1	**48.5**	**62.4**	38.7	**48.5**	75.0	**40.8**	**52.9**	28.5	**59.8**	24.1	**47.9**	**55.3**	**88.5**	66.7
$\mathcal{L}_D^{dec}$	89.1	64.7	28.5	62.4	29.7	54.3	30.9	67.7	79.7	27.8	35.9	51.9	46.2	52.6	40.0	47.0	**77.3**	29.9	35.6	**33.3**	40.0	23.0	42.5	47.4	87.1	59.3
$\mathcal{L}_D^{SPKD\text{-}avg}$	88.9	64.6	29.6	68.6	35.8	64.9	76.8	76.8	74.5	31.0	16.8	57.1	46.2	55.1	26.1	34.0	77.0	34.5	33.9	33.1	56.4	25.2	42.1	50.0	87.8	61.8
$M_0(0-10)$	95.3	86.4	34.4	85.6	69.7	79.3	94.6	87.6	93.1	44.2	91.9	78.4	-	-	-	-	-	-	-	-	-	-	-	78.4	96.1	90.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	74.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	70.5	72.1	73.6	93.9	84.2

classes is slightly lower since less training samples are available for such classes. In this case the gain of mIoU is smaller: no single method is able to get a large improvement, however $\mathcal{L}_D^{enc}$ and $\mathcal{L}_D^{SPKD\text{-}avg}$ are the best performing approaches. A critical example is the behavior of the *sheep* class, whose accuracy is highly reduced after the addition of the correlated *cow* class.

As a final experiment, we investigate the sequential addition of the last 10 classes, *i.e.*, $M_{10}(11 \rightarrow 20)$: the results of per-class IoU are shown in Table 3.11. In this case the best strategy is to apply distillation on the intermediate feature space ($\mathcal{L}_D^{enc}$), that allows to improve the mIoU of 18.2% with respect to the standard scheme achieving quite good performance in this extreme case. The result is even more effective because the proposed method outperforms the standard approach by 11.9% of mIoU on new classes and by 24.1% on old ones.
Again, we can confirm that correlated classes highly influence each other results: for example the addition of the *sheep* class (low IoU and high pixel accuracy) leads to a large degradation of the performance on the *cow* class (low IoU and low pixel accuracy). The same happens to the *bus* class when *train* is added. This phenomenon is mitigated by the proposed modifications which improve the IoU of the *cow* class from 7.8% of the baseline to 55.1% of the best proposed method and the IoU of the *bus* class from 1.5% to 74.0%.

3.5 Experimental Results on MSRC-v2

Finally, we briefly show the per-class IoU results obtained on the MSRC-v2 dataset. We sort the classes according to their occurrence inside the dataset and we perform three experiments:

Table 3.12: Per-class IoU of the proposed approaches on MSRC-v2 when the last class, *i.e.*, *boat*, is added

$M_1(20)$	grass	building	sky	road	tree	water	book	car	cow	bicycle	flower	body	sheep	sign	face	cat	chair	aeroplane	dog	bird	**mIoU old**	boat	**mIoU**	**mPA**	**mCA**
Fine-tuning	93.4	79.9	92.9	70.8	86.7	79.3	94.5	93.2	86.8	91.9	95.4	83.0	84.0	91.3	87.9	79.5	89.1	78.3	74.0	74.3	85.3	51.2	83.7	92.4	89.6
E_F	94.3	82.2	92.7	58.8	86.2	72.7	98.5	91.3	89.0	89.2	97.7	84.1	89.7	93.3	88.0	93.6	94.7	85.3	89.5	78.3	87.5	56.0	86.0	92.5	91.7
$\mathcal{L}_D^{cls\text{-}T}$	94.7	82.8	93.5	83.4	87.5	87.4	98.1	94.4	89.3	91.5	97.8	83.2	88.2	93.9	86.0	84.2	96.2	85.2	80.0	82.6	89.0	59.3	87.6	94.5	92.9
E_F, $\mathcal{L}_D^{cls\text{-}T}$	94.8	83.4	93.4	81.7	87.3	86.6	98.6	92.7	89.4	89.8	97.9	83.9	89.5	93.8	87.3	94.2	96.4	86.3	91.0	83.5	**90.1**	58.2	**88.6**	**94.6**	**93.7**
$\mathcal{L}_D^{enc}$	94.9	82.3	92.7	77.9	87.5	84.4	98.5	92.3	88.6	90.4	97.7	84.5	87.5	92.8	88.6	93.6	97.3	84.5	88.3	81.2	89.3	54.6	87.6	94.1	92.8
$\mathcal{L}_D^{dec}$	94.8	83.7	94.1	80.8	88.1	84.6	97.5	90.8	86.6	88.2	97.9	84.9	82.8	95.4	88.6	76.1	94.6	83.3	71.2	81.3	87.2	40.8	85.0	93.9	91.1
$\mathcal{L}_D^{SPKD\text{-}avg}$	94.3	81.9	93.2	63.5	87.2	74.5	98.7	91.2	88.9	89.6	97.9	85.9	87.8	91.0	91.4	74.0	92.7	82.2	68.0	79.9	85.7	**66.9**	84.8	92.4	91.1
$M_0(0-19)$	94.9	84.7	93.3	88.7	88.9	90.9	98.6	94.4	89.0	89.6	98.0	83.7	89.6	94.1	86.7	95.0	95.4	86.2	92.4	82.1	90.8	-	90.8	95.5	96.0
$M_0(0-20)$	94.8	82.3	94.6	87.3	88.8	92.5	98.6	94.1	90.9	89.7	98.0	87.4	92.0	91.2	89.9	93.9	95.9	84.8	90.3	87.3	86.7	75.1	90.5	95.4	95.5

Table 3.13: Per-class IoU of the proposed approaches on MSRC-v2 when 5 classes are added at once.

$M_1(16-20)$	grass	building	sky	road	tree	water	book	car	cow	bicycle	flower	body	sheep	sign	face	cat	**mIoU old**	chair	aeroplane	dog	bird	boat	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	91.9	79.5	93.9	83.9	84.1	88.5	96.0	94.0	49.0	91.6	93.8	82.9	59.3	87.3	90.4	37.5	81.5	86.3	74.9	43.6	66.4	63.5	67.0	78.0	91.1	85.2
E_F	93.8	80.3	92.0	83.1	86.0	84.1	97.9	92.3	81.0	87.7	97.1	64.8	78.2	96.5	89.1	59.0	85.2	89.4	70.1	52.2	66.8	53.8	66.5	80.7	92.1	86.2
$\mathcal{L}_D^{cls\text{-}T}$	93.5	81.0	91.5	81.9	86.6	84.8	98.4	94.2	86.3	90.5	96.5	77.3	81.8	95.8	89.6	69.8	87.5	91.0	78.1	69.0	69.9	67.6	75.1	84.5	93.3	**90.8**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	94.4	80.7	91.6	83.2	86.7	83.5	98.8	92.6	80.7	88.5	97.0	66.3	78.8	96.1	89.2	58.8	85.4	89.7	67.4	52.8	67.4	54.5	66.4	80.9	92.4	87.6
$\mathcal{L}_D^{enc}$	93.3	80.4	92.8	85.6	87.5	88.9	98.7	94.0	72.1	89.1	95.7	80.8	81.4	91.2	86.4	57.2	85.9	87.7	72.5	62.4	71.1	66.5	72.0	82.6	93.1	88.5
$\mathcal{L}_D^{dec}$	94.8	81.6	93.4	85.4	87.8	88.0	95.8	94.0	87.4	87.8	96.0	84.9	83.1	91.5	89.0	68.2	**88.0**	88.3	81.4	70.1	78.5	59.8	**75.6**	**85.1**	**93.9**	**90.8**
$\mathcal{L}_D^{SPKD\text{-}avg}$	92.3	79.5	93.5	83.5	85.7	88.9	95.7	94.0	73.8	91.7	93.9	84.4	72.8	87.9	90.0	45.6	84.6	84.2	67.8	55.1	71.4	59.1	67.5	80.5	92.2	86.7
$M_0(0-15)$	94.2	83.5	89.7	85.7	88.9	88.3	98.9	94.5	88.8	89.0	98.0	85.0	91.6	95.1	89.8	96.4	91.1	-	-	-	-	-	-	91.1	95.2	96.0
$M_0(0-20)$	94.8	82.3	94.6	87.3	88.8	92.5	98.6	94.1	90.9	89.7	98.0	87.4	92.0	91.2	89.9	93.9	91.6	95.9	84.8	90.3	87.3	75.1	86.7	90.5	95.4	95.5

we add the last class, the last 5 in a single shot or the last 5 sequentially. The results are shown in Tables 3.12, 3.13 and 3.14.

Table 3.12 shows the results for the addition of the last class, *i.e.*, *boat*. The average accuracy is higher on this dataset, however also in this case the fine-tuning approach is surpassed by all the proposed methods, proving the effectiveness of the distillation strategies. As already noticed in Table 3.1, the best method when adding only one class is $M_1(20)[\mathcal{L}_D^{cls\text{-}T}, E_F]$. The best strategy to learn the new class, instead, is $M_1(20)[\mathcal{L}_D^{SPKD\text{-}avg}]$ which significantly outperforms all the other methods. A qualitative example is shown in the first row of Figure 3.7, where we can notice how the fine-tuning tends to find *boat* samples close to the water, while no boat is present in this image. This artifact is reduced by $\mathcal{L}_D^{cls\text{-}T}$ and $\mathcal{L}_D^{enc}$ and completely removed by $\mathcal{L}_D^{dec}$ and $\mathcal{L}_D^{SPKD\text{-}avg}$.

The second scenario regards the addition of the last five classes at once (Table 3.13). Here the best strategy is $M_1(16-20)[\mathcal{L}_D^{dec}]$ with an overall gap of 7.1% of mIoU. The most challenging aspect on this dataset is the recognition of new classes in place of visually similar old ones; especially the *cat* and *cow* classes are frequently exchanged for the newly introduced *dog* class. For example, the pixel accuracy of the cat raises from 37.5% of fine-tuning to 71% for the best proposed approach. On this dataset the classes appear mainly alone or with few other classes in the images, thus it is less likely to observe the effects of the correlation between classes belonging to similar contexts. In the visual example in second row of Figure 3.7 the fine-tuning approach misleads the *cat* as a *dog* (which is among the classes being added). The issue is mitigated by the proposed strategies that are able to detect at least part of the object as a *cat*.

As third experiment, we consider the sequential addition of five classes in Table 3.14. Here

Table 3.14: Per-class IoU of the proposed approaches on MSRC-v2 when 5 classes are added sequentially.

$M_5(16 \rightarrow 20)$	grass	building	sky	road	tree	water	book	car	cow	bicycle	flower	body	sheep	sign	face	cat	**mIoU old**	chair	aeroplane	dog	bird	boat	**miou inc.**	**mIoU**	**mPA**	**mCA**
Fine-tuning	86.3	71.8	93.4	69.4	77.1	67.0	90.7	71.8	53.0	88.7	79.6	69.1	49.6	74.8	90.3	34.5	72.9	34.2	51.6	35.6	17.7	35.5	34.9	63.9	83.8	73.2
E_F	88.6	74.4	93.2	72.1	80.7	74.3	94.9	90.0	70.4	88.9	90.3	68.5	64.1	85.3	87.1	59.4	80.1	57.2	51.1	30.1	23.5	36.3	39.6	70.5	87.2	76.3
$\mathcal{L}_D^{cls\text{-}T}$	92.7	78.9	90.7	74.5	86.2	71.6	97.8	93.7	88.7	90.8	95.5	67.7	77.4	93.4	88.2	64.9	84.5	**83.4**	**69.8**	**47.2**	41.3	**54.7**	**59.3**	**78.5**	90.7	**86.9**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	92.9	79.5	90.4	81.2	85.3	81.2	99.0	93.3	82.2	88.7	96.0	67.6	67.5	92.8	87.6	79.5	85.3	82.0	50.3	29.7	35.1	40.2	47.5	76.3	90.6	84.8
$\mathcal{L}_D^{enc}$	91.5	78.7	91.5	75.8	86.6	77.9	98.6	94.1	84.6	89.8	97.0	80.1	74.6	87.1	86.3	78.0	85.8	75.3	46.8	30.7	43.2	31.7	45.5	76.2	90.5	83.3
$\mathcal{L}_D^{dec}$	93.2	81.9	93.5	79.3	87.4	77.4	98.2	90.4	86.4	85.0	95.1	78.1	70.7	93.2	84.2	85.1	**86.2**	41.6	61.7	40.9	**68.3**	16.2	45.7	76.6	**91.1**	84.5
$\mathcal{L}_D^{SPKD\text{-}avg}$	91.2	78.4	94.7	62.0	81.1	64.8	96.4	91.1	22.6	88.6	94.2	82.5	47.2	85.0	91.0	30.0	75.0	55.3	51.4	36.2	16.9	42.0	40.3	66.8	85.4	75.6
$M_0(0-15)$	94.2	83.5	89.7	85.7	88.9	88.3	98.9	94.5	88.8	89.0	98.0	85.0	91.6	95.1	89.8	96.4	91.1	-	-	-	-	-	-	91.1	95.2	96.0
$M_0(0-20)$	94.8	82.3	94.6	87.3	88.8	92.5	98.6	94.1	90.9	89.7	98.0	87.4	92.0	91.2	89.9	93.9	91.6	95.9	84.8	90.3	87.3	75.1	86.7	90.5	95.4	95.5

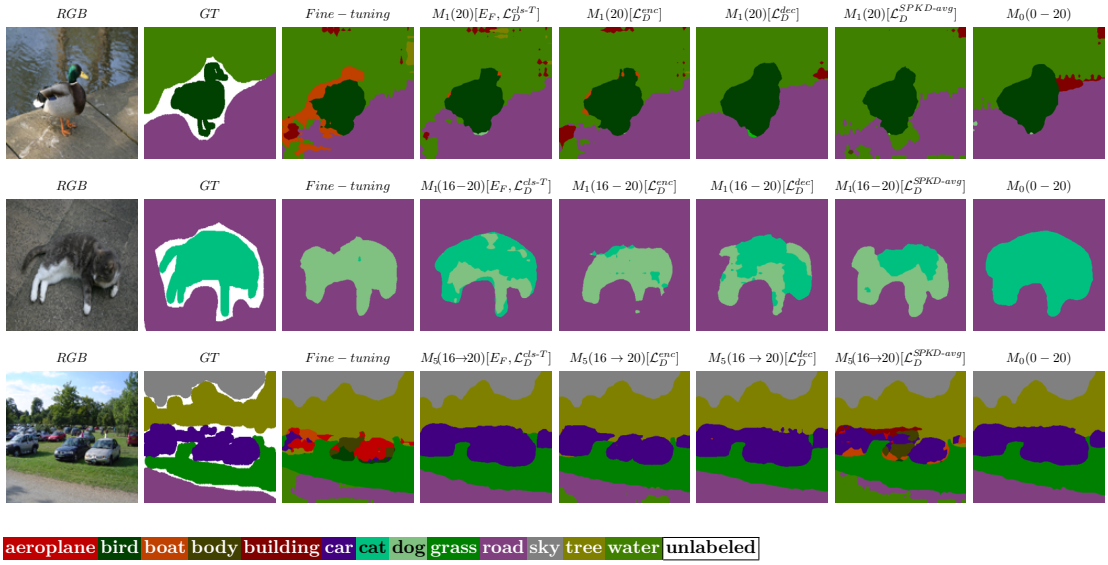


Figure 3.7: Qualitative results on sample scenes on MSRC-v2. The first row regards the addition of the last class (*i.e.*, *boat*), the second row regards the addition of the last 5 classes at once, the third row regards the addition of the last 5 classes sequentially. The classes added are respectively *chair*, *aeroplane*, *dog*, *bird*, *boat*.

the drop in accuracy is larger and the performance on some newly introduced classes are poor due to the limited number of samples in this smaller dataset. We can appreciate how the best proposed strategies, $\mathcal{L}_D^{cls\text{-}T}$ and $\mathcal{L}_D^{dec}$, are able to largely outperform the fine-tuning approach by a huge margin. In the visual example in row 3 of Figure 3.7, the *cars* are misled with *boats* and *aeroplanes* when fine-tuning, while the proposed strategies almost completely solve the problem.

3.6 Ablation Studies

In this section we discuss some ablation studies of the proposed framework. First, we run some preliminary experiments on the disjoint experimental setting; then, we analyze the influence of backbone pre-training; finally, we report an ablation study on multi-layer knowledge distillation.

Table 3.15: Ablation study comparing ImageNet (mIoU$_I$) and MSCOCO (mIoU$_M$) pre-training on VOC2012.

	$M_1(20)$		$M_1(11-20)$		$M_{10}(11 \rightarrow 20)$	
	mIoU$_I$	mIoU$_M$	mIoU$_I$	mIoU$_M$	mIoU$_I$	mIoU$_M$
Fine-tuning	62.8	66.5	64.3	65.6	35.9	37.1
E_F	68.8	71	62.2	64.1	50.9	54.5
$\mathcal{L}_D^{cls\text{-}T}$	68.9	71.5	**65.5**	**66.3**	36.5	37.5
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**70.9**	**71.8**	63.7	64.3	54.0	54.9
$\mathcal{L}_D^{enc}$	70.6	71.5	65.2	66	**54.5**	**55.3**
$\mathcal{L}_D^{dec}$	68.1	70.0	64.8	65.7	46.3	47.4
$\mathcal{L}_D^{SPKD\text{-}avg}$	68.8	71.0	64.5	65.6	48.8	50

Table 3.16: Difference of pre-training strategies in incremental learning on VOC2012 in terms of mIoU when the last class, *i.e.*, the *tv/monitor* class, is added.

$M_1(20)$	ImageNet pre-training			MSCOCO pre-training		
	mIoU old	miou inc.	mIoU	mIoU old	miou inc.	mIoU
Fine-tuning	65.0	19.6	62.8	66.6	63.8	66.5
E_F	70.2	40.5	68.8	71.5	61.9	71.0
$\mathcal{L}_D^{cls\text{-}T}$	70.2	43.7	68.9	71.6	**68.2**	71.5
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**71.9**	**51.5**	**70.9**	**72.2**	64.2	**71.8**
$\mathcal{L}_D^{enc}$	71.6	50.1	70.6	71.9	62.3	71.5
$\mathcal{L}_D^{dec}$	69.7	35.2	68.1	70.1	67.5	70.0
$\mathcal{L}_D^{SPKD\text{-}avg}$	70.0	44.6	68.8	71.1	67.6	71.0
$M_0(0-19)$	71.3	-	71.3	73.0	-	73.0
$M_0(0-20)$	71.6	68.4	71.4	73.3	78.7	73.6

3.6.1 Backbone Pre-Training

As we observed, semantic segmentation architectures are typically composed of an encoding and a decoding stage. The encoder is trained to learn useful but compact representations of the scene which are then processed by the decoder to produce the output segmentation map with a classification score for each pixel. Such architectures are highly complex and the weights of the encoder are always pre-trained on very large datasets: *e.g.*, on ImageNet [121] or MSCOCO [120]. For incremental learning strategies it is important to ensure that the pre-training does not affect the incremental steps since it could contain information concerning novel classes to be learned in the incremental steps.

The ideal strategy would be to avoid the pre-training at all, but this is necessary since the Pascal VOC2012 dataset is too small to train the network from scratch in a reliable way.

To better investigate this issue, *i.e.*, to ensure that the information learned during the pre-training is not affecting the incremental learning results, we performed a set of experiments using a different pre-training done on the ImageNet dataset (that has only classification annotations without any segmentation information).

The comparison in terms of mIoU is shown in Table 3.15 for the addition of the last class, of the last 10 classes at once and of the last 10 classes sequentially. Pre-training on ImageNet leads to a slightly lower starting mIoU for all the scnarios, however the improvements obtained by the proposed strategies are similar to those with MSCOCO pre-training and so are their final rankings (*i.e.*, the best approaches in the various scenarios are the same). Additionally, also the gap with respect to the fine-tuning approach is coherent between the two pre-training strategies.

More detailed results are shown in Tables 3.16, 3.17 and 3.18 respectively for the addition of the last class, of the last 10 classes at once and of the last 10 classes sequentially. The new pre-training leads to a slightly lower starting mIoU, however the improvements obtained by the proposed strategies are similar and so are the final rankings of the methods. For example, in the

Table 3.17: Difference of pre-training strategies in incremental learning of the proposed approaches on VOC2012 in terms of mIoU when 10 classes are added at once.

$M_1(11-20)$	ImageNet pre-training			MSCOCO pre-training		
	mIoU old	miou inc.	mIoU	mIoU old	miou inc.	mIoU
Fine-tuning	65.8	62.7	64.3	67.5	63.5	65.6
E_F	67.5	56.4	62.2	69.4	58.2	64.1
$\mathcal{L}_D^{cls\text{-}T}$	67.8	**62.9**	**65.5**	69.1	63.3	**66.3**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**69.0**	57.8	63.7	**69.6**	58.5	64.3
$\mathcal{L}_D^{enc}$	67.7	62.4	65.2	68.4	63.3	66.0
$\mathcal{L}_D^{dec}$	67.3	62.0	64.8	68.1	63.1	65.7
$\mathcal{L}_D^{SPKD\text{-}avg}$	66.0	**62.9**	64.5	67.6	**63.4**	65.6
$M_0(0-10)$	77.2	-	77.2	78.4	-	78.4
$M_0(0-20)$	71.6	71.2	71.4	74.9	72.1	73.6

Table 3.18: Difference of pre-training strategies in incremental learning of the proposed approaches on VOC2012 in terms of mIoU when 10 classes are added sequentially.

$M_{10}(11\rightarrow 20)$	ImageNet pre-training			MSCOCO pre-training		
	mIoU old	miou inc.	mIoU	mIoU old	miou inc.	mIoU
Fine-tuning	36.5	35.2	35.9	38.0	36.0	37.1
E_F	60.1	40.8	50.9	**63.1**	44.9	54.4
$\mathcal{L}_D^{cls\text{-}T}$	37.4	35.6	36.5	38.5	36.4	37.5
E_F, $\mathcal{L}_D^{cls\text{-}T}$	62.5	44.6	54.0	63.7	45.2	54.9
$\mathcal{L}_D^{enc}$	**61.4**	**46.9**	**54.5**	62.1	**47.9**	**55.3**
$\mathcal{L}_D^{dec}$	50.9	41.3	46.3	51.9	42.5	47.4
$\mathcal{L}_D^{SPKD\text{-}avg}$	55.8	41.2	48.8	57.1	42.1	50.0
$M_0(0-10)$	77.2	-	77.2	78.4	-	78.4
$M_0(0-20)$	71.6	71.2	71.4	74.9	72.1	73.6

sequential addition of 10 classes (Table 3.18) the best approach is in both cases $\mathcal{L}_D^{enc}$ and the difference in terms of mIoU between the old and the new pre-training is only 0.8%. The same gap is also present between the most performing method in Table 3.17 ($\mathcal{L}_D^{cls\text{-}T}$) when applied to the different pre-trained networks.

To further highlight the effect of the pre-training strategy we show some relative results in Table 3.19. In particular, we show the difference of mIoU for each method between the two pre-training strategies (Δ_{M-I}). Then, we report the difference of mIoU between each proposed method and fine-tuning when using the two pre-training schemes ($\Delta_{FT,I}$ when using ImageNet and $\Delta_{FT,M}$ when using MSCOCO). From these results it is clear that the ranking of the methods remains always the same and that the gaps are coherent. Although MSCOCO data consist in a better pre-training for the segmentation task, the same relative analysis holds for both scenarios.

In conclusion, using a different pre-training does not change the effectiveness of the proposed strategies and the general message of the work remains unaltered, even if the starting point can be a little different.

3.6.2 Experimental Analyses on Disjoint Setup on VOC2012

As we observed in Section 2.4, in some applications we may be interested in performing incremental steps with images where only the new classes are annotated, and previously seen classes are set as background (*i.e.*, the so-called disjoint setup). This scenario could be extremely helpful to save time and resources on the annotation of previous classes.

Interestingly, the problem could be tackled with the same set of proposed techniques, which prove to be of quite general application. Nevertheless, the difference with respect to the se-

Table 3.19: Ablation of the different pre-training strategies on ImageNet (I) and on MSCOCO (M). Δ_{M-I}: difference of mIoU between the two pre-training. $\Delta_{FT,I}$: difference of mIoU between each proposed method and fine-tuning in case ImageNet is used as pre-training.

	$M_1(20)$			$M_1(11-20)$			$M_{10}(11 \rightarrow 20)$		
	Δ_{M-I}	$\Delta_{FT,I}$	$\Delta_{FT,M}$	Δ_{M-I}	$\Delta_{FT,I}$	$\Delta_{FT,M}$	Δ_{M-I}	$\Delta_{FT,I}$	$\Delta_{FT,M}$
Fine-tuning	3.7	0.0	0.0	1.3	0.0	0.0	1.2	0.0	0.0
E_F	2.2	6.0	4.5	1.9	-2.1	-1.5	3.6	15.0	17.4
$\mathcal{L}_D^{cls\text{-}T}$	2.6	6.1	5.0	0.8	1.2	0.7	1.0	0.6	0.4
E_F, $\mathcal{L}_D^{cls\text{-}T}$	0.9	8.1	5.3	0.6	-0.6	-1.3	0.9	18.1	17.8
$\mathcal{L}_D^{enc}$	0.9	7.8	5.0	0.8	0.9	0.4	0.8	18.6	18.2
$\mathcal{L}_D^{dec}$	1.9	5.3	3.5	0.9	0.5	0.1	1.1	10.4	10.3
$\mathcal{L}_D^{SPKD\text{-}avg}$	2.2	6.0	4.5	1.1	0.2	0.0	1.2	12.9	12.9

Table 3.20: Results of the proposed approaches on the disjoint setup of VOC2012 in terms of mIoU when the last class, *i.e.*, the *tv/monitor*, is added.

$M_1(20)$	mIoU old	miou inc.	mIoU
Fine-tuning	65.8	18.2	63.5
E_F	71.1	36.5	69.4
$\mathcal{L}_D^{cls\text{-}T}$	69.4	29.5	67.5
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**72.0**	44.9	**70.7**
$\mathcal{L}_D^{enc}$	71.6	**48.3**	**70.7**
$\mathcal{L}_D^{dec}$	70.5	32.8	68.7
$\mathcal{L}_D^{SPKD\text{-}avg}$	70.7	43.0	69.4
$M_0(0-19)$	73.4	-	73.4
$M_0(0-20)$	73.7	70.5	73.6

Table 3.21: Results of the proposed approaches on the disjoint setup of VOC2012 in terms of mIoU when 10 classes are added at once.

$M_1(11-20)$	mIoU old	miou inc.	mIoU
Fine-tuning	65.8	**63.0**	64.4
E_F	67.9	58.5	63.4
$\mathcal{L}_D^{cls\text{-}T}$	68.5	**63.0**	**65.9**
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**68.8**	58.8	64.0
$\mathcal{L}_D^{enc}$	67.1	62.7	65.0
$\mathcal{L}_D^{dec}$	67.5	62.2	64.9
$\mathcal{L}_D^{SPKD\text{-}avg}$	65.9	62.7	64.4
$M_0(0-10)$	78.4	-	78.4
$M_0(0-20)$	74.9	72.1	73.6

Table 3.22: Results of the proposed approaches on the disjoint setup of VOC2012 in terms of mIoU when 10 classes are added sequentially.

$M_{10}(11 \rightarrow 20)$	mIoU old	miou inc.	mIoU
Fine-tuning	47.9	44.0	46.0
E_F	56.8	45.5	51.4
$\mathcal{L}_D^{cls\text{-}T}$	50.6	45.0	47.9
E_F, $\mathcal{L}_D^{cls\text{-}T}$	**62.8**	46.6	55.1
$\mathcal{L}_D^{enc}$	62.2	**50.7**	**56.7**
$\mathcal{L}_D^{dec}$	51.3	42.7	47.2
$\mathcal{L}_D^{SPKD\text{-}avg}$	56.4	42.5	49.8
$M_0(0-10)$	78.4	-	78.4
$M_0(0-20)$	74.9	72.1	73.6

quential incremental learning protocol evaluated up to this point is at least twofold: first, the background class changes distribution at every incremental step; second, less information is present in the new images.

We analyze this experimental protocol in three cases: namely, the addition of the last class (in Table 3.20), the addition of the last 10 classes at once (in Table 3.21), and the addition of the last 10 classes sequentially one at a time (in Table 3.22).

In general, we can observe that, as expected, the final mIoU results are typically lower than in the previous experimental setting. However, it is important to notice that the ranking of the proposed methods remains substantially the same as in the sequential scenario: *i.e.*, E_F, $\mathcal{L}_D^{cls\text{-}T}$ achieves a mIoU of 70.7% and outperforms the other methods when adding the last class (compare Table 3.20 with Table 3.1); $\mathcal{L}_D^{cls\text{-}T}$ outperforms the other proposals with a mIoU of 65.9% when adding 10 classes at once (compare Table 3.21 with Table 3.4); finally $\mathcal{L}_D^{enc}$ performs best in the sequential addition of 10 classes with a mIoU of 56.7% (compare Table 3.22 with Table 3.11).

3.6.3 Ablation on Multi-Layer Knowledge Distillation

In this section we report an ablation study on multi-layer knowledge distillation, *i.e.*, on the impact of applying distillation at different stages in the network.

The results are reported in Table 3.23: first of all applying feature-level distillation at the end of each block of the ResNet-101 encoder (called "$\mathcal{L}_D^{enc}$ on 5 blocks of ResNet-101" in the table), leads to results in between $\mathcal{L}_D^{enc}$ and E_F, as we may expect, since the approach is constraining the features at different resolutions of the encoder but does not completely freeze them.

Moving to multi-layer distillation at the decoder, we can appreciate how distilling early layers

Table 3.23: Ablation study on multi-layer knowledge distillation in incremental learning on VOC2012 in terms of mIoU when 10 classes are added sequentially.

$M_{10}(11 \rightarrow 20)$	mIoU old	miou inc.	mIoU
E_F	63.1	44.9	54.4
$\mathcal{L}_D^{enc}$	62.1	47.9	55.3
$\mathcal{L}_D^{dec}$	51.9	42.5	47.4
$\mathcal{L}_D^{enc}$ on 5 blocks of ResNet-101	63.0	45.8	54.8
$\mathcal{L}_D^{dec}$ on layer 1 only	58.8	48.7	54.0
$\mathcal{L}_D^{dec}$ on layer 4 only	45.6	36.8	41.4
$\mathcal{L}_D^{dec}$ on layers 1 and 2 only	54.4	47.5	51.1
$\mathcal{L}_D^{dec}$ on layers 3 and 4 only	47.2	39.2	43.4
$M_0(0-10)$	78.4	-	78.4
$M_0(0-20)$	74.9	72.1	73.6

of the decoding stage (*e.g.*,"$\mathcal{L}_D^{dec}$ on layer 1 only" and "$\mathcal{L}_D^{dec}$ on layers 1 and 2 only") pushes the results toward distillation on the intermediate features (*i.e.*, $\mathcal{L}_D^{enc}$). On the other hand, distilling the last layers of the decoding stage (*e.g.*, "$\mathcal{L}_D^{dec}$ on layer 4 only" and "$\mathcal{L}_D^{dec}$ on layers 3 and 4 only"), as expected, pushes the results toward distillation on the output layer (*i.e.*, $\mathcal{L}_D^{cls\text{-}T}$).

3.7 Summary

In this chapter we have explored many knowledge distillation techniques applied to continual semantic segmentation, combined with a standard cross-entropy loss to optimize the performance on new classes while preserving at the same time high accuracy on old ones. The proposed method does not need any stored image regarding the previous sets of classes making it suitable for applications with strict privacy and storage requirements. Additionally, only the previous model is used to update the current one, thus reducing the memory consumption.

In the following, we briefly summarize the main achievements of each proposed strategy highlighting the best solutions and the challenges related to the various scenarios. First, we have shown that fine-tuning always leads to catastrophic forgetting of old classes and prevents learning new ones. Freezing the whole encoder (E_F) or its first couple of layers (E_{2LF}) lead to similar results and such methods are especially effective (even more if used in combination with other strategies) when a few classes are added to the model, as the frozen encoder fails to accommodate large changes in the input distribution. The first loss we propose (*i.e.*, $\mathcal{L}_D^{cls\text{-}T}$) is quite general and it achieves the highest results when a few incremental steps are made, but it suffers over multiple iterations. The second loss we propose (*i.e.*, $\mathcal{L}_D^{enc}$) is often among the best performing approaches and it is especially useful when dealing with multiple incremental stages thanks to the preservation of the feature space. To further improve the organization of the decoding features, we designed $\mathcal{L}_D^{dec}$, which has proven to be useful when many classes are added at a time. Finally, in an attempt to improve the distillation from the intermediate layers we considered the loss $\mathcal{L}_D^{SPKD\text{-}avg}$ working on the activation functions. This method robustly achieves higher results with respect to fine-tuning in all scenarios performing similarly to the other distillation methods.

Extensive experiments on Pascal VOC2012 and MSRC-v2 datasets showed that the proposed methodologies are able to largely outperform the fine-tuning approach, where no additional provisions are exploited. However, continual semantic segmentation is a novel task with still a lot of space for improvement, as proved by the gap from the results achieved by the same network architecture with a single-step training, *i.e.*, when all training examples are available and employed at the same time. We argue that the main reason of such performance gap has

to be identified with the entanglement of the latent space, as we observed in the examples in Figure 3.6: features of previously seen classes are associated in the subsequent training steps to features of novel visually-similar classes. Feature-level regularization and disentanglement, indeed, will be the main focus of the next chapter.

4

Latent-Space Regularization of the Learned Embeddings

4.1 Introduction

This chapter investigates and analyzes regularization methods for the latent representations learned by the deep neural network models.

Differently from the majority of previous approaches both in image classification [48–50] and semantic segmentation [18, 23, 54, 55], we do not mainly or solely rely on output-level knowledge distillation. In this chapter, we explore the latent space organization which has been only marginally investigated in the current literature, and we empirically prove it to be compatible to other existing techniques. The methodologies described in the following, indeed, can serve as complementary techniques of those already proposed at the output space. The main idea is depicted in Figure 4.1, where some of the latent space constraints are introduced. First, a prototype matching is devised to enforce features extraction consistency on old classes between the cumulative prototype computed using all previous samples and the current prototype (*i.e.*, the prototype computed on the current batch only). In other words, we force the encoder to produce similar latent representations for previously seen classes in the new steps. Second, a features sparsification constraint makes room in the latent space to accommodate novel classes. To further regularize the latent space, we introduce an attraction-repulsion rule similar in spirit to the recent advancements in contrastive learning. Finally, to enforce the decoder to preserve discriminability on previous categories during classification, we employ a targeted output-level distillation.

4.1.1 Background

Latent Space Organization. The analysis of the latent space organization is becoming crucial towards understanding and improvement of classification models [104, 105]. Recently, some attention has been devoted to latent regularization in continual image classification [106, 107, 126]. Besides this, one of the emerging paradigms is constrastive learning applied to visual representations. Dating back to [127], these approaches learn representations by contrasting positive against negative pairs and have been recently re-discovered for deep learning. Many works use a memory bank to store the instance class representation vector [128–133], while some others

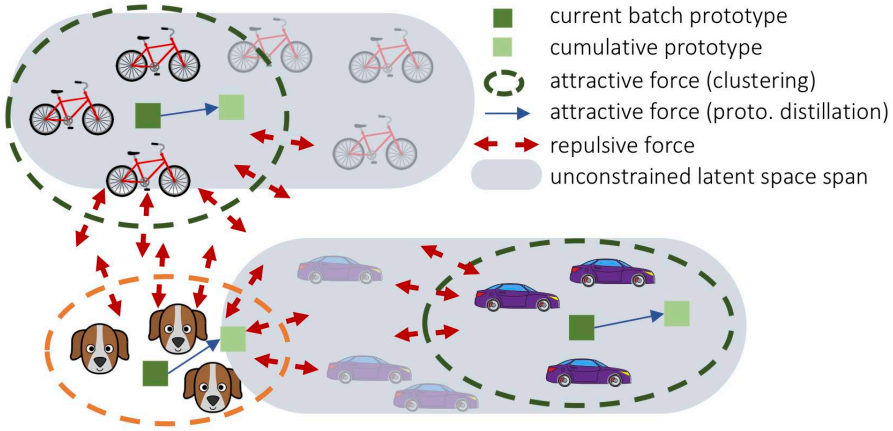


Figure 4.1: Our continual learning scheme is driven by three main components: latent contrastive learning, prototypes matching and features sparsity. Latent representations of old classes are preserved via prototypes matching and clustering, whilst also making room for accommodating new classes via sparsity and repulsive force of contrastive learning. The decoder preserves previous knowledge via output-level distillation. In the figure, bike and cars represent old classes and leave more space to new classes (the dog) thanks to the novel constraints (green dotted ovals versus gray-filled ovals).

explore the usage of in-batch negative samples instead [134–137]. The contrastive learning objective proposed in this work moves from opposition of positive and negative pairs and also recalls features clustering (if features belong to the same class) and separation (if features belong to different classes), which has been recently applied to adapt semantic segmentation models across domains [31, 138, 139].
Prototype-based regularizing terms gained a great interest and, in particular, have been largely used in the literature of few-shot learning [140–142], to learn prototypical representations of each category, and domain adaptation, to enforce orthogonality [143, 144] or centroid matching [145, 146].
Finally, to minimize the interference among features we drive them to be channel-wise sparse. Only limited attention has been given on sparsity for deep learning architectures [126]; however, some prior techniques exist for domain adaptation on linear models exploiting sparse codes on a shared dictionary between the domains [147, 148].

As we have already mentioned in Section 2.4, although continual semantic segmentation has only been faced recently, it already comes with different experimental protocols depending on how the incremental data are considered: namely, *sequential* (new images are labeled with both new and old classes), *disjoint* (new images are labeled with only new classes, old classes are assigned to the background) and *overlapped* (new images are labeled with only new classes, images are repeated across training steps with different semantic maps associated to them). In this section, we devise a common framework which allows to tackle all these scenarios and can be applied in combination with previous techniques, which has never been attempted before. We evaluate on standard semantic segmentation datasets, like Pascal VOC2012 [114] and ADE20K [117], in many experimental scenarios.

4.1.2 Contributions

The proposed continual learning scheme shapes the latent space to reduce forgetting whilst improving the recognition of novel classes. Our framework is driven by three novel components which we also combine on top of existing techniques effortlessly. First, prototypes matching en-

forces latent space consistency on old classes, constraining the encoder to produce similar latent representation for previously seen classes in the subsequent steps. Second, features sparsification allows to make room in the latent space to accommodate novel classes. Finally, contrastive learning is employed to cluster features according to their semantics while tearing apart those of different classes. Extensive evaluation on the Pascal VOC2012 and ADE20K datasets demonstrates the effectiveness of our approach, significantly outperforming state-of-the-art methods.

Our work is the first combining together contrastive learning, sparsity and prototypes matching to regularize latent space for segmenting new categories over time.

Summing up, the main contributions of this work are:

1. we investigate class-incremental learning in semantic segmentation, providing a common framework for different experimental protocols;

2. we explore the latent space organization and we propose complementary techniques with respect to the existing ones;

3. we propose novel knowledge preservation techniques based on prototypes matching, contrastive learning and features sparsity;

4. we benchmark our approach on standard semantic segmentation datasets outperforming state-of-the-art continual learning methods.

4.2 Problem Definition and Setups

In the following, we denote the input image space with $\mathcal{X} \in \mathbb{R}^{H \times W \times 3}$ with spatial dimensions H and W, the set of classes (or categories) with $\mathcal{C} = \{c_i\}_{i=0}^{C-1}$ and the output space with $\mathcal{Y} \in \mathcal{C}^{H \times W}$ (*i.e.*, the segmentation map). Given a training set $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, where $(\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$, we aim at finding a map M that solves the semantic segmentation problem, from the input space to a pixel-wise class probability vector $M : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times C}$. Then, the output segmentation mask is computed as $\hat{\mathbf{y}}_n = \arg\max_{c \in \mathcal{C}} M(\mathbf{x}_n)[h, w, c]$, where $h = 1, .., H$, $w = 1, ..., W$ and $M(\mathbf{x}_n)[h, w, c]$ is the probability for class c in pixel (h, w). Nowadays, M is typically some auto-encoder model made by an encoder E and a decoder D (*i.e.*, $M = E \circ D$). We call $\mathbf{F}_n = E(\mathbf{x}_n)$ the feature map of $\mathbf{x}_n$, and $\mathbf{y}_n^*$ the downsampled segmentation map matching the spatial dimensions of $\mathbf{F}_n$.

In the standard supervised setting it is assumed that the training set $\mathcal{T}$ is available at once and the model is learned in one shot. In the continual learning scenario, instead, training is achieved over multiple iterations each carrying a novel category to learn and a subset of the training data. More formally, at each learning step k the previous label set $\mathcal{C}_{k-1}$ is expanded with a set of novel classes $\mathcal{S}_k$ forming a new label set $\mathcal{C}_k = \mathcal{C}_{k-1} \cup \mathcal{S}_k$. Additionally, a new training subset $\mathcal{T}_k \subset \mathcal{X} \times \mathcal{C}_k$ is made available and used to update the previous model into a new model M_k. Step $k = 0$ consists of a standard supervised training performed with only a subset of training data and classes. As in the standard incremental class learning scenario, we assume the different sets of new classes to be disjoint with the exception of the peculiar background class c_0, *i.e.*, $\mathcal{S}_i \cap \mathcal{S}_j = \{c_0\}$.

4.3 Method

In this section, we provide a detailed description of the core modules of the proposed method. Our approach leverages a contrastive learning objective applied over the feature representations,

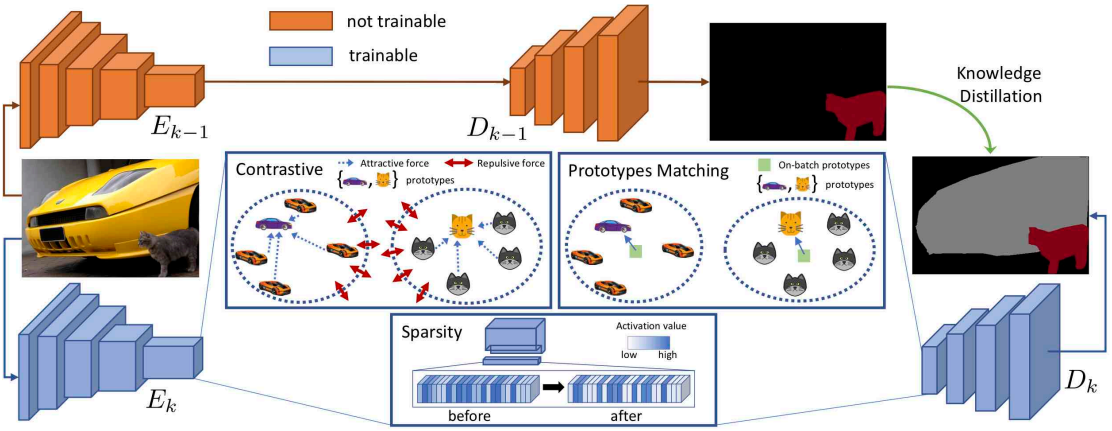


Figure 4.2: Overview of the proposed approach, with an old class (*cat*) and a new class (*car*). Latent representations of old classes are preserved over time via prototypes matching and clustering, whilst also making room for accommodating new classes via sparsity and repulsive force in contrastive learning. The decoder is constrained to act as in previous steps on previous classes via output-level distillation.

with novel prototypes matching and sparsity constraints. Specifically, features repulsion and attraction based on the semantic classes are enforced by grouping together features of the same class, while simultaneously pushing away those of different categories. We further regularize the distribution of latent representations by the joint application of prototypes matching and sparsity. While prototypes matching seeks for an invariant representation of the features extracted for the old classes, the sparsity objective encourages a lower volume of active feature channels from latent representations (*i.e.*, it concentrates the energy of features on few dimensions) to free up space for new classes.

An overall scheme of our approach is shown in Figure 4.2: the training objective is given by the combination of a cross-entropy loss ($\mathcal{L}_{ce}$) with the proposed modules. $\mathcal{L}_{ce}$ is the usual cross-entropy loss for all the classes except for the background. The ground truth of the background, indeed, is not directly compared with its probabilities, but with the probability of having either an old class or the background in the current model [23]. Formally, at step k the background probabilities $M(\mathbf{x}_n)[h, w, c_0]$ are replaced by $\sum_{c \in \mathcal{C}_{k-1}} M(\mathbf{x}_n)[h, w, c]$. The rationale behind this is that the background class could incorporate statistics of previous classes in both the disjoint and overlapped protocols.

The other components are a prototypes matching target ($\mathcal{L}_{pm}$), a contrastive learning objective ($\mathcal{L}_{cl}$) and a sparsity constraint ($\mathcal{L}_{sp}$), which will be detailed in the following sections. The training objective is then computed as:

$$\mathcal{L}'_{tot} = \mathcal{L}_{ce} + \lambda_{pm} \cdot \mathcal{L}_{pm} + \lambda_{cl} \cdot \mathcal{L}_{cl} + \lambda_{sp} \cdot \mathcal{L}_{sp} \tag{4.1}$$

where the λ parameters balance the multiple losses and have been tuned using a validation set (see Section 4.4). Our aim is to seek for disentangled latent representations characterized by semantic-driven regularization and to show that this approach can achieve comparable or superior results with respect to standard regularization methods (*e.g.*, output-level knowledge distillation). We further integrate the proposed framework with an output-level knowledge distillation objective [20] and we show that its effect is highly not overlapping, achieving increased accuracy. The training objective comprising an unbiased output-level distillation module is

defined as:

$$\mathcal{L}_{tot} = \mathcal{L}'_{tot} + \lambda_{kd} \cdot \mathcal{L}_{kd} \tag{4.2}$$

4.3.1 Prototypes Matching

Prototypes (*i.e.*, class-centroids) are vectors that are representative of each category that appears in the dataset. During training, the features extracted by the encoder contribute in forming the latent prototypical representation of each class. To preserve the geometrical structure of the features of old classes we apply prototypes matching. Current prototypes $\hat{\mathbf{p}}_c$ (*i.e.*, computed on the current batch of images) are forced to be placed close to their representation learned from the previous steps $\mathbf{p}_c$. We use the Frobenius norm $||\cdot||_F$ as metric distance [141,149,150]. More formally:

$$\mathcal{L}_{pm} = \frac{1}{|\mathcal{C}_{k-1}|}||\mathbf{p}_c - \hat{\mathbf{p}}_c||_F \quad c \in \mathcal{C}_{k-1} \tag{4.3}$$

The prototypes are computed in-place with a running average updated at each training step with supervision. At training step t with batch $\mathcal{B}$ of B images, the prototypes are updated for a generic class c as:

$$\mathbf{p}_c[t] = \frac{1}{Bt}\left(B(t-1)\mathbf{p}_c[t-1] + \sum_{\mathbf{x}_n \in \mathcal{B}} \frac{\sum_{\mathbf{f}_i \in \mathbf{F}_n} \mathbf{f}_i \mathbb{1}[y_i^* = c]}{|\mathbb{1}\left[\mathbf{y}_n^* = c\right]|}\right) \tag{4.4}$$

initialized to $\mathbf{p}_c[0] = \mathbf{0}\ \forall c$. $\mathbf{f}_i \in \mathbf{F}_n$ is a generic feature vector and y_i^* the corresponding pixel in $\mathbf{y}_n^*$, $\mathbb{1}\left[\mathbf{y}_n^* = c\right]$ indicates the pixels in $\mathbf{y}_n^*$ associated to c and $|\cdot|$ denotes cardinality.

We update the prototypes only when we have ground truth labels for that class to avoid incorporating the mutable statistics of the background class: we exclude the background from the incremental steps in the disjoint protocol (as it could contain old classes) and in the overlapped scenario (as it could contain old and future classes).

For the current batch $\mathcal{B}$ of an incremental training stage, the current (or in-batch) prototypes $\hat{\mathbf{p}}_c[t]$ are computed as:

$$\hat{\mathbf{p}}_c[t] = \frac{1}{B} \sum_{\mathbf{x}_n \in \mathcal{B}} \begin{cases} \frac{\sum_{\mathbf{f}_i \in \mathbf{F}_n} \mathbf{f}_i \mathbb{1}[y_i^* = c]}{|\mathbb{1}[\mathbf{y}_n^* = c]|} & \text{if sequential} \\ \frac{\sum_{\mathbf{f}_i \in \mathbf{F}_n} \mathbf{f}_i \mathbb{1}[\hat{z}_i^* = c]}{|\mathbb{1}[\hat{\mathbf{z}}_n^* = c]|} & \text{otherwise} \end{cases} \tag{4.5}$$

where $\hat{\mathbf{z}}_n^*$ (with pixels $\hat{z}_i^*$) is a pseudo-labeled segmentation map computed from the ground truth data by replacing the background region with the prediction from the previous model, since in the disjoint and overlapped protocols old classes are labeled as background. The difference between (4.4) and (4.5) lies in the usage of pseudo-labels: we use them in (4.5) to compute prototypes for old classes in the current batch since we may not have any label for them, but we avoid to use them in (4.4), since there is no need to update prototypes computed using the ground truth at previous steps with data from less reliable pseudo-labels.

4.3.2 Contrastive Learning

The second component is similar to recent contrastive learning [130,133] and clustering [138,139] approaches to constraint the latent space organization. The underlying idea is to structure the latent space in order to have features of the same category clustered near their prototype and at the same time to force prototypes to be far one from the other. We argue that this organization helps also in continual learning to mitigate forgetting and to facilitate the addition of novel

classes, as features are clustered and there is more separation between the clusters. In formal terms, the constraint is defined by a loss $\mathcal{L}_{cl}$ made of an attractive term $\mathcal{L}_{cl}^{a}$ and a repulsive term $\mathcal{L}_{cl}^{r}$, as follows:

$$\mathcal{L}_{cl}^{a} = \frac{1}{|c_j \in \mathbf{y}_n^*|} \sum_{c_j \in \mathbf{y}_n^*} \sum_{\mathbf{f}_i \in \mathbf{F_n}} \| (\mathbf{f}_i - \mathbf{p}_{c_j}) \mathbb{1}[y_i^* = c_j] \|_F \tag{4.6}$$

$$\mathcal{L}_{cl}^{r} = \frac{1}{|c_j \in \mathbf{y}_n^*|} \sum_{c_j \in \mathbf{y}_n^*} \sum_{\substack{c_k \in \mathbf{y}_n^* \\ c_k \neq c_j}} \frac{1}{\| \hat{\mathbf{p}}_{c_j} - \hat{\mathbf{p}}_{c_k} \|_F} \tag{4.7}$$

The objective is composed of two terms: $\mathcal{L}_{cl}^{a}$ measures how close features are from their respective centroids and $\mathcal{L}_{cl}^{r}$ how spaced out prototypes corresponding to different semantic classes are. Hence, the effect provided by the loss minimization is twofold: firstly, feature vectors from the same class are tightened around class feature centroids; secondly, features from separate classes are subject to a repulsive force applied to feature centroids, moving them apart.

4.3.3 Features Sparsity

To enforce the regularizing effect brought by contrastive learning, we introduce a further feature-wise objective on the latent space. We propose a sparsity loss to decrease the number of active feature channels of latent vectors. First, to give the same importance to all classes, we normalize each feature vector with respect to the maximum value any of the feature channels for that particular class assumes, *i.e.*:

$$\bar{\mathbf{f}}_i = \frac{\mathbf{f}_i}{\max_{\substack{g_{j,l} \in \mathbf{g}_j \\ y_j^* = y_i^*}} g_{j,l}} \qquad \mathbf{f}_i, \mathbf{g}_j \in \mathbf{F}_n \tag{4.8}$$

We design the sparsity constraint as the ratio between a stretching function (we used the sum of exponentials) and a linear function (*i.e.*, the sum) applied over each feature vector, which is minimized when the energy is concentrated in a few channels (since the normalized features assume values ≤ 1). The sparsity constraint is thus defined as:

$$\mathcal{L}_{sp} = \frac{1}{|\mathbf{f}_i \in \mathbf{F}_n|} \sum_{\mathbf{f}_i \in \mathbf{F}_n} \frac{\sum_j \exp\left(\bar{f}_{i,j}\right)}{\sum_j \bar{f}_{i,j}} \tag{4.9}$$

While the contrastive learning objective forces features to lie within tight semantically-consistent well-distanced clusters, the sparsity constraint aims at narrowing the set of active channels with the aim of letting room for the representation of upcoming classes. In other words, by constraining features of the same classes to be tightly clustered and to be spaced apart from features of other classes and sparse, we can preserve geometrical space (few active channels) and expressiveness (division in well-separated clusters) for the latent representation of future classes. Empirically, we found entropy-based minimization methods in the latent space [151] to be less reliable for our task. In Section 4.7 we show how to handle degenerate cases of (4.9) and an ablation on other sparsifying strategies.

4.3.4 Output-Level Knowledge Distillation

The last component of this method is an output-level knowledge distillation which we show to be complementary to the previously introduced strategies. Indeed, we add knowledge distillation on top of all the other components to transfer knowledge from the old model's classifier to the current one. While previous constraints regularize the latent space achieving simultaneously an invariant features extraction with respect to previous steps and an easier addition of novel categories, output-level knowledge distillation directly acts on the classifier, to preserve its discriminative ability regarding old classes. In particular, we start from the preliminary considerations of [18,20] and we employ the unbiased distillation proposed in [23] as natural extension to the case in which the background may contain other categories. In this case we avoid to re-normalize the probabilities from the previous step and, instead, we compare the background probability from the previous step with the probability of having either a new class or the background (this accounts for the fact that the background in the previous steps may include samples of the new classes, see [23]).

4.4 Training Procedure

To train and benchmark our approach we resort to two publicly available datasets (the Pascal VOC2012 and the ADE20K) following [18, 20, 23, 51].

The proposed strategy is agnostic to the backbone architecture. For the experimental evaluation of all the compared methods we use a standard DeepLab-v3+ [4] architecture with ResNet-101 [152] as backbone (differently from [23] for wider reproducibility) with output stride of 16. The backbone has been initialized using a pre-trained model on ImageNet [121] (see Section 4.6 for a detailed discussion of the impact of different pre-training strategies). We optimize the network weights following [6] with SGD and with same learning rate policy, momentum and weight decay. The first learning step involves an initial learning rate of 10^{-2}, which is decreased to 10^{-3} for the following steps as done in [23, 51]. The learning rate is decreased with a polynomial decay rule with power 0.9. In each learning step we train the models with a batch size of 8 for 30 epochs for Pascal VOC2012 and a batch size of 4 for 60 epochs for ADE20K. Following [6], we crop the images to 512×512 during both training and validation and we apply the same data augmentation (*i.e.*, random scaling the input images of a factor from 0.5 to 2.0 and random left-right flipping during training). In order to set the hyper-parameters of each method, we follow the same continual learning protocol of [23, 60], i.e, we used 20% of the training set as validation and we report the results on the original validation set of the datasets. We use Pytorch to develop and train all the models on a NVIDIA 2080 Ti GPU. The code is available at: `https://lttm.dei.unipd.it/paper_data/SDR/`.

4.5 Experimental Results

We evaluate the performance of our method (denoted in the tables with **SDR**, *i.e.*, Sparse and Disentangled Representations) against some state-of-the-art continual learning frameworks. We report as a lower limit the performance of the naïve fine-tuning approach (FT), which consists in training the model on the newly available training data with no additional provisions, while the upper limit is given by the offline single-shot training (offline) on the whole dataset $\mathcal{T}$ and on all the classes at once. Then, we compare with 3 recent continual semantic segmentation schemes: ILT [18], which combines latent and output level knowledge distillation, CIL [55], which adds class importance weighting to output-level knowledge distillation, and MiB [23], which deals

Table 4.1: mIoU on multiple incremental scenarios and protocols on VOC2012. Best in **bold**, runner-up underlined. †: results from [23].

	19-1									15-5									15-1								
	sequential			disjoint			overlapped			sequential			disjoint			overlapped			sequential			disjoint			overlapped		
Method	old	new	all	old	new	all	old	new	all	old	new	all	old	new	all	old	new	all	old	new	all	old	new	all	old	new	all
FT	63.4	21.2	61.4	35.2	13.2	34.2	34.7	14.9	33.8	62.0	38.1	56.3	8.4	33.5	14.4	12.5	36.9	18.3	49.0	17.8	41.6	5.8	4.9	5.6	4.9	3.2	4.5
LwF [49]	67.2	26.4	65.3	65.8	28.3	64.0	62.6	23.4	60.8	68.0	43.0	62.1	39.7	33.3	38.2	67.0	41.8	61.0	33.7	13.7	29.0	26.2	15.1	23.6	24.0	15.0	21.9
LwF-MC [48]	49.2	0.9	46.9	38.5	1.0	36.7	37.1	2.3	35.4	70.6	19.5	58.4	41.5	25.4	37.6	59.8	22.6	51.0	12.1	1.9	9.7	6.9	2.1	5.7	6.9	2.3	5.8
ILT [18]	64.3	22.7	62.3	66.9	23.4	64.8	50.2	29.2	49.2	71.3	**47.8**	65.7	31.5	25.1	30.0	69.0	46.4	63.6	49.2	**30.3**	**48.3**	6.7	1.2	5.4	5.7	1.0	4.6
CIL [55]	64.1	22.8	62.1	62.6	18.1	60.5	35.1	13.8	34.0	63.8	39.8	58.1	42.6	35.0	40.8	14.9	37.3	20.2	52.4	22.3	45.2	33.3	**15.9**	29.1	6.3	4.5	5.9
MiB† [23]	-	-	-	69.6	25.6	67.4	70.2	22.1	67.8	-	-	-	71.8	43.3	64.7	75.5	49.4	69.0	-	-	-	46.2	12.9	37.9	35.1	13.5	29.7
MiB [23]	68.2	31.9	66.5	67.0	26.0	65.1	69.6	23.8	67.4	73.0	44.4	66.1	47.5	34.1	44.3	73.1	44.5	66.3	35.7	11.0	29.8	39.0	15.0	33.3	44.5	11.7	36.7
SDR (ours)	68.4	**35.3**	66.8	69.9	**37.3**	68.4	69.1	**32.6**	67.4	73.6	46.7	67.2	73.5	**47.3**	67.2	75.4	**52.6**	69.9	**58.5**	10.1	47.0	59.2	12.9	48.1	44.7	**21.8**	39.2
SDR + MiB	**70.6**	24.8	**68.5**	**70.8**	31.4	**68.9**	**71.3**	23.4	**69.0**	**74.6**	43.8	**67.3**	**74.6**	44.1	**67.3**	**76.3**	50.2	**70.1**	58.1	11.8	47.1	**59.4**	14.3	**48.7**	**47.3**	14.7	**39.5**
offline	75.5	73.5	75.4	75.5	73.5	75.4	75.5	73.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4

with the background distribution shift and proposes an unbiased weight initialization rule. We also report the results on LwF [49] (together with its single-headed version LwF-MC [48]), that according to [23] is the best performing continual image classification algorithm when adapted to semantic segmentation. For a fair comparison, all the methods have been re-trained with a standard DeepLab-v3+ [4] architecture with ResNet-101 [152] as backbone.

4.5.1 Pascal VOC2012

Following previous works [18, 20, 23, 51], we design three main experiments adding one class (19-1), five classes at once (15-5) and five classes sequentially (15-1) added in alphabetical order. In Table 4.1 we report comprehensive results on the three experimental protocols defined in Section 2.4. Results are averaged for mIoU of classes in the base step (*old*), for classes in the incremental steps (*new*) and for *all* classes, and are reported at the end of all the incremental steps. For [23] we also report the original results in their paper (denoted with MiB†), that uses a different backbone (thus different pre-trained model) and batch size.

We can appreciate forgetting of previous classes and intransigence in learning new ones even when adding as little as one class (the *tv/monitor* class is added) in the scenario 19-1. FT always leads to the worst mIoU in terms of *old*, *new* and *all* classes. Incremental methods designed for semantic segmentation allow for a stable improvement across the experimental protocols, in particular MiB, that is specifically targeted to solve the disjoint and the overlapped scenarios, while CIL and ILT encounter difficulties in the overlapped scenario. Also LwF allows for a good improvement while its single-headed version has lower performance in this scenario. Our method (SDR) significantly outperforms all the competitors in the disjoint and overlapped scenarios (with a gap of more than 3% against the best competing approach in the disjoint setup), while in the sequential setup the gap is smaller. Further adding on top of our method the MiB framework (*i.e.*, unbiased cross entropy, knowledge distillation and classifier initialization), which we regard as the current state-of-the-art approach for class incremental semantic segmentation, the results increase on all the scenarios, showing that proposed techniques are complementary with respect to previous schemes.

When moving to the addition of 5 classes at once (*i.e.*, *potted plant*, *sheep*, *sofa*, *train*, *tv/monitor*) we immediately notice an overall increased drop of performance of all compared methods, especially in disjoint and overlapped protocols, due to the increased domain shift occurring when adding more classes at once with very variable content. In this and in the following scenario, indeed, we are adding to the model classes belonging to different macroscopic groups, according to [114], which are responsible for a variegate distribution: three indoor classes (*potted plant*, *sofa* and *tv/monitor*), one animal class (*sheep*) and one vehicle class (*train*). All compared methods obtain a relevant improvement with respect to FT but are always surpassed by SDR,

which in particular outrun the best competing method (MiB) by more than 20% in the disjoint scenario.

In the final scenario we add the last 5 classes sequentially in 5 consecutive learning steps. This approach leads to the largest accuracy drop being the model exposed to a reiterated addition of single classes, which are also coming from different semantic contexts. In the sequential scenario LwF and MiB (which is designed for background shift) show poor final accuracy. ILT and CIL, instead, show results comparable to our approach. In the disjoint and in the overlapped scenarios all the methods heavily suffer from the semantic shift undergone by the background class: LwF (both versions) and ILT have poor performance in these scenarios, while CIL is able to achieve some improvement only in the disjoint scenario. The best competitor is again MiB that is able to obtain a mIoU of 33% and 36.7% in the disjoint and overlapped scenarios respectively. Our approach (SDR) is able to significantly increase the final mIoU in both scenarios to 48.1% and 39.2%; it achieves a remarkable result especially in the disjoint scenario thanks to the novel features-level constraints which help the model to maintain accuracy on old classes while learning new ones.

Visual results for each scenario are shown in Figures 4.3, 4.4 and 4.5, respectively for sequential, disjoint and overlapped protocols. In each figure, 3 images for each scenario (*i.e.*, 19-1, 15-5 and 15-1) are depicted. We compare our method with naïve fine tuning and the competitors, *i.e.*, LwF [49], ILT [18], CIL [55] and MiB [23]. The images show how our approach is able to alleviate forgetting and at the same time accommodate for new classes to learn. On the other side, the fine-tuning and the compared approaches often deviate (*i.e.*, are biased) in predicting novel classes being added or the special background class.

In Figure 4.3 our method segments better the shape of the *horse* in row 1, it does not overfit to predict the *sofa* class (one of the classes being added) in row 5, and properly individuates the correct objects in rows 8 and 9.

In Figure 4.4 our method does not mislead the *bus* windows with *tv/monitor* instances (row 1) differently from several competitors (which are more biased toward predicting the novel class), and it is the only one able to distinguish the *sheep* in row 6 and the *tv/monitor* in row 8.

In Figure 4.5 our method outperforms competitors in detecting the shape of the objects in rows 1 and 3, it is not biased towards novel added classes in rows 4 and 5 (*bus* and *cow* are correctly detected, while competitors place newly seen visually-similar classes *train* and *sheep*), and can better segment the shapes of the *horse*, *dog* and *airplane* in rows 7, 8 and 9.

4.5.2 ADE20K

Following [23], we split the dataset into disjoint image sets with the only constraint that a minimum number of images (*i.e.*, 50) have labeled pixels on $\mathcal{C}_k$. Classes are ordered according to [117]. In this comparison we report the same competing methods of Section 4.5.1. The scenarios we consider are the addition of the last 50 classes at once (100-50), of the last 50 classes 10 at a time (100-10) and of the last 100 classes in 2 steps of 50 classes each (50-50). The results are summarized in Table 4.2, where we can appreciate that the proposed approach outperforms competitors in every scenario, in particular with a larger gain when multiple incremental steps are performed. When adding 50 classes at a time LwF-MC and CIL achieve low results and are outperformed by the other competitors (*i.e.*, LwF, ILT and MiB), which in turn are always consistently surpassed by our framework. In the scenario 100-10, instead, all competing approaches (except for MiB) are unable to provide useful outputs leading to extremely low results, while our method stands out from competitors outperforming also MiB by a good margin.

Visual results are shown in Figure 4.6, which confirm our considerations showing how SDR produces less noisy predictions and does not overestimate the background as some competitors.

Figure 4.3: Qualitative results on sample scenes in different scenarios (19-1, 15-5 and 15-1) on Pascal VOC2012 of the proposed method and of competing approaches in the sequential setup.

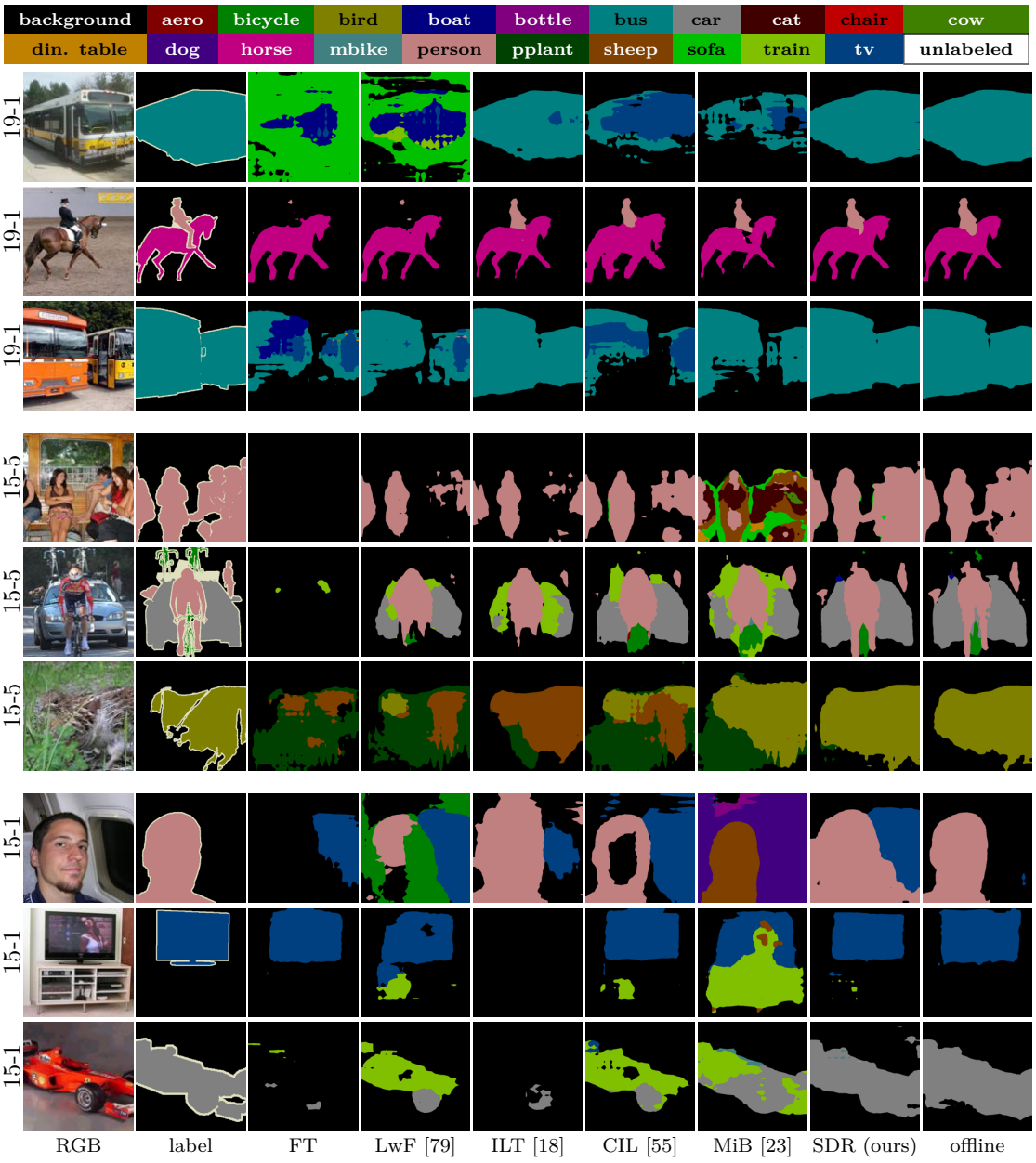


Figure 4.4: Qualitative results on sample scenes in different scenarios (19-1, 15-5 and 15-1) on Pascal VOC2012 of the proposed method and of competing approaches in the disjoint setup.

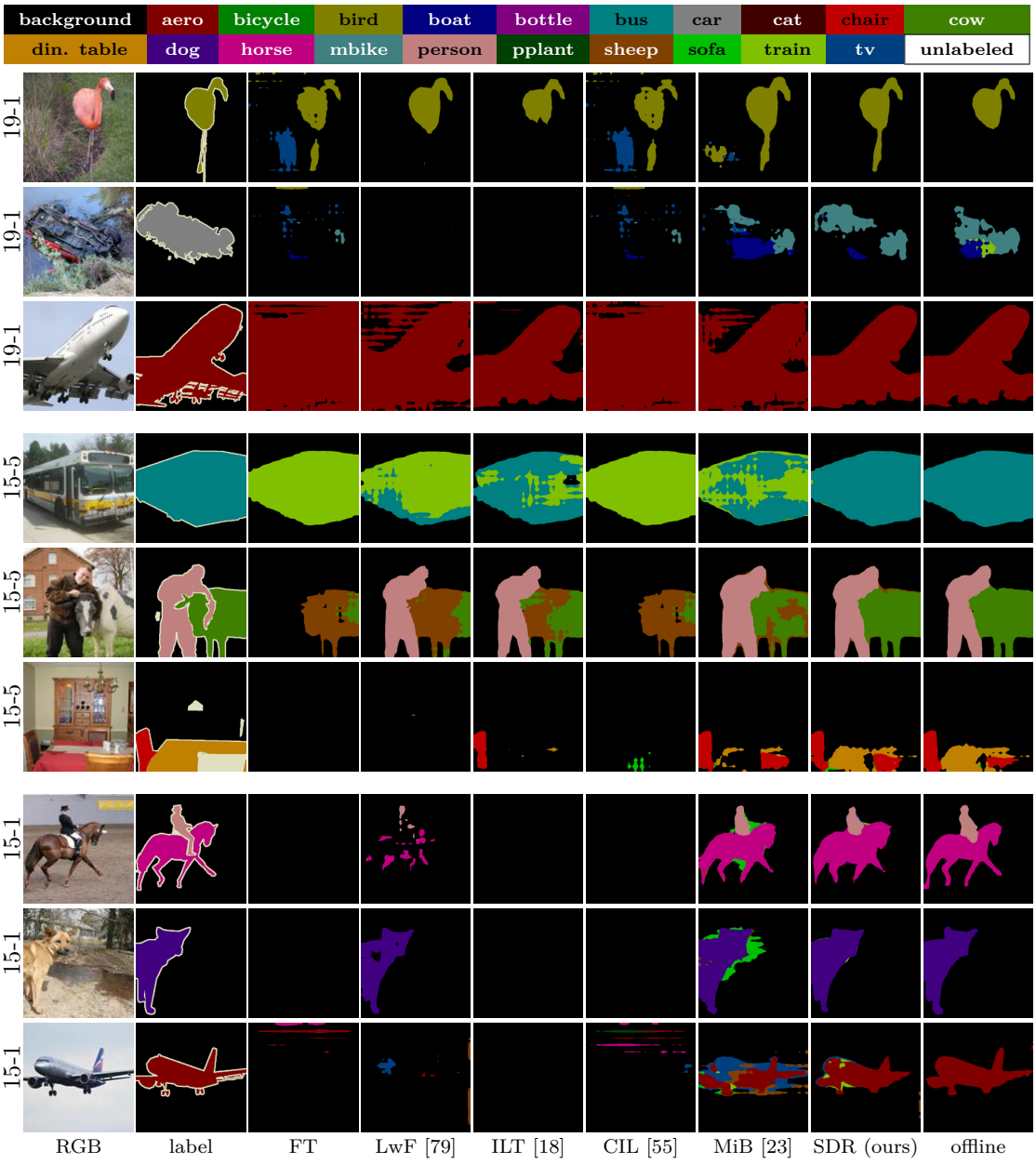


Figure 4.5: Qualitative results on sample scenes in different scenarios (19-1, 15-5 and 15-1) on Pascal VOC2012 of the proposed method and of competing approaches in the overlapped setup.

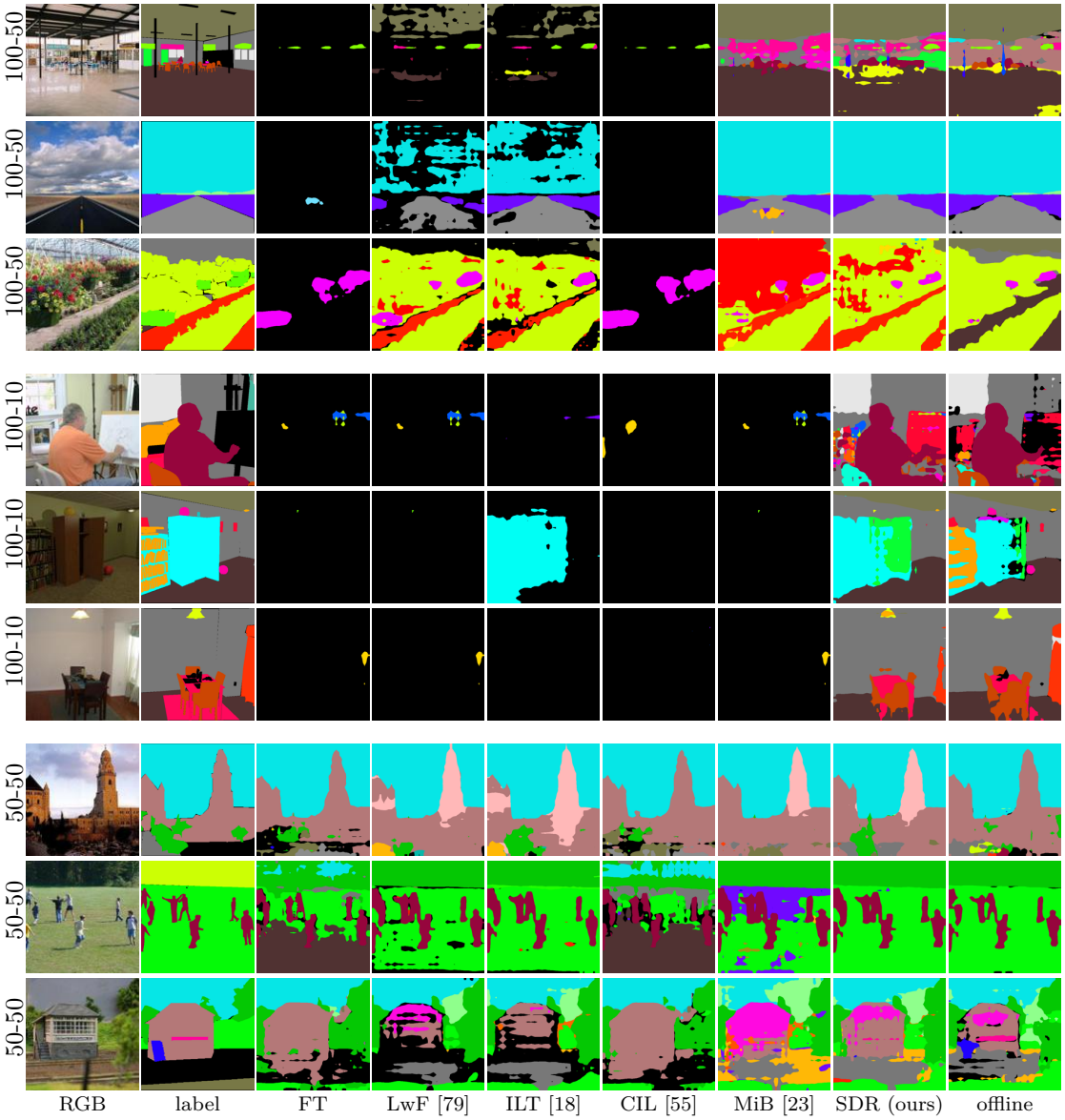


Figure 4.6: Qualitative results on sample scenes in different scenarios (100-50, 100-10 and 50-50) on ADE20K of the proposed method and of competing approaches.

Table 4.2: mIoU over multiple incremental scenarios on disjoint setup of ADE20K. Best in **bold**, runner-up underlined.

	100-50			100-10			50-50		
Method	old	new	all	old	new	all	old	new	all
FT	0.0	22.5	7.5	0.0	2.5	0.8	13.9	12.0	12.6
LwF [49]	25.0	23.9	24.6	5.4	5.6	5.5	32.2	22.9	26.0
LwF-MC [48]	8.6	0.0	5.8	0.0	0.9	0.3	2.8	0.5	1.2
ILT [18]	27.2	21.7	25.4	0.0	0.2	0.8	41.9	21.1	28.0
CIL [55]	0.0	22.5	7.5	0.0	2.0	0.6	14.0	11.9	12.6
MiB [23]	**37.6**	24.7	33.3	21.0	5.3	15.8	39.1	22.6	28.1
SDR (ours)	37.4	24.8	33.2	**28.9**	7.4	21.7	40.9	23.8	29.5
SDR+MiB	37.5	**25.5**	**33.5**	**28.9**	**11.7**	**23.2**	**42.9**	**25.4**	**31.3**
offline	43.9	27.2	38.3	43.9	27.2	38.3	50.9	32.1	38.3

In particular, we show 3 images for each scenario (*i.e.*, 100-50, 100-10, 50-50). Again, we can appreciate how our method largely outperforms compared approaches in all scenarios better capturing the details of the shapes of the objects (e.g, in rows 1-4) and not degenerating into an overestimation of the background (*e.g.*, in the 100-10 scenario). In particular, we notice how compared approaches have big difficulties in handling multiple additions of multiple classes (they struggle in tackling catastrophic forgetting in the 100-10 scenario), while our method can achieve reasonably good output segmentation maps also in the most challenging scenarios.

4.5.3 Qualitative Results Across Incremental Steps

In this section we analyze the performance across the various incremental steps, comparing our method with the top performing competitor (*i.e.*, MiB [23]).

Pascal VOC2012. The results on two sample scenes from this dataset are reported in Figure 4.7 for the disjoint 15-1 experimental protocol, where an initial training stage over 15 classes is followed by 5 incremental learning steps each carrying one class to be learned. In the first row our method shows quite robust results across the different learning steps, being able to preserve content semantics. MiB, instead, is able to avoid catastrophic forgetting for one incremental step but it degenerates after introducing the *sheep* class (step 2), which is predicted in spite of *person* probably due to the confusion of the arms and legs (caused also by their similar color). The latent representations got even more damaged across subsequent steps, while our approach (SDR) is able to reduce the interference on latent representations of old classes. Similar considerations also holds for the second set of images, although in this scenario forgetting is less evident: our approach is able to achieve superior performance thanks to correct spatial localization and latent disentanglement.

ADE20K. For this dataset we consider two distinct scenarios: *i.e.*, 5 incremental steps each adding 10 categories to the model (100-10) in Figure 4.8, and 2 incremental steps each adding 50 classes to the model (50-50) in Figure 4.9.

The first scenario is definitely the most challenging one as the model need to adapt 5 times to discover new (and possibly unrelated) classes. Nevertheless, we can appreciate that our model obtain quite robust results across the various steps in the 2 sample scenes shown in Figure 4.8, while MiB suffers more from catastrophic forgetting previous knowledge. In the first sample scene our approach shows a small gradual degradation across the multiple steps, while MiB firstly completely looses the wall on the background in step 2, then the curtain in step 3 and finally also the hand basin in step 4. Similarly, in the second scene our approach maintains very good results across all the steps, while MiB at the third step misleads the sky on the background.

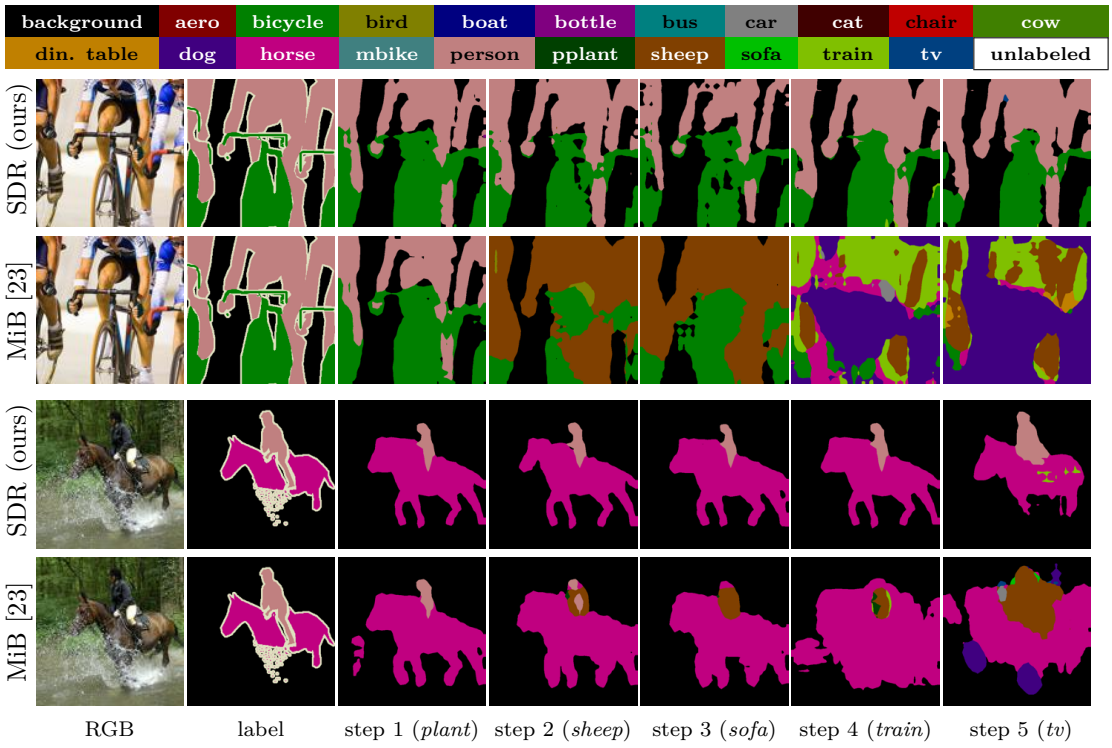


Figure 4.7: Qualitative results on sample scenes in the disjoint experimental protocol 15-1 on Pascal VOC2012 during the various incremental steps.

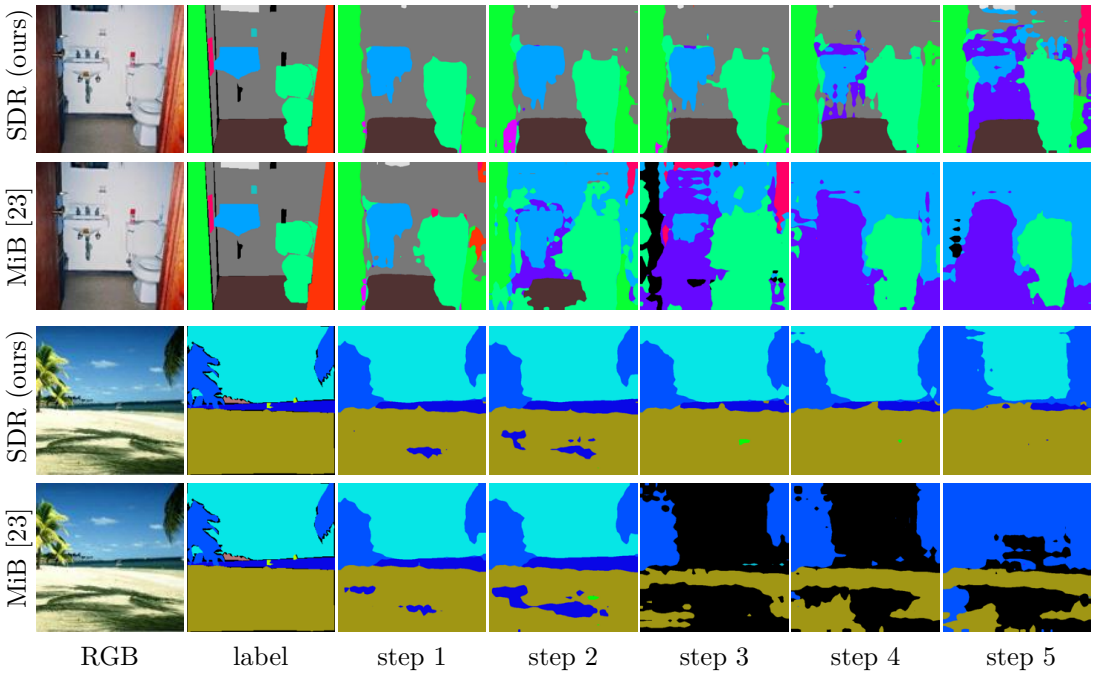


Figure 4.8: Qualitative results on sample scenes in experimental protocol 100-10 on ADE20K during the various incremental steps.

In Figure 4.9 we consider the case in which only two incremental steps with 50 classes each are performed. It can be appreciated how in the first step the predicted segmentation maps are quite precise according to both our approach and MiB, but, in both examples, MiB produces a less precise map after the second incremental step. More in detail, we remark some differences: our model can identify the tree (green) in the first image, that MiB only partially captures in the first step and completely misses it in the second. Similarly, SDR preserves the walls (gray) in the second image that MiB misleads in the second step. Again, the latent space regularization helps in preserving previous classes representations and in accommodating new classes.

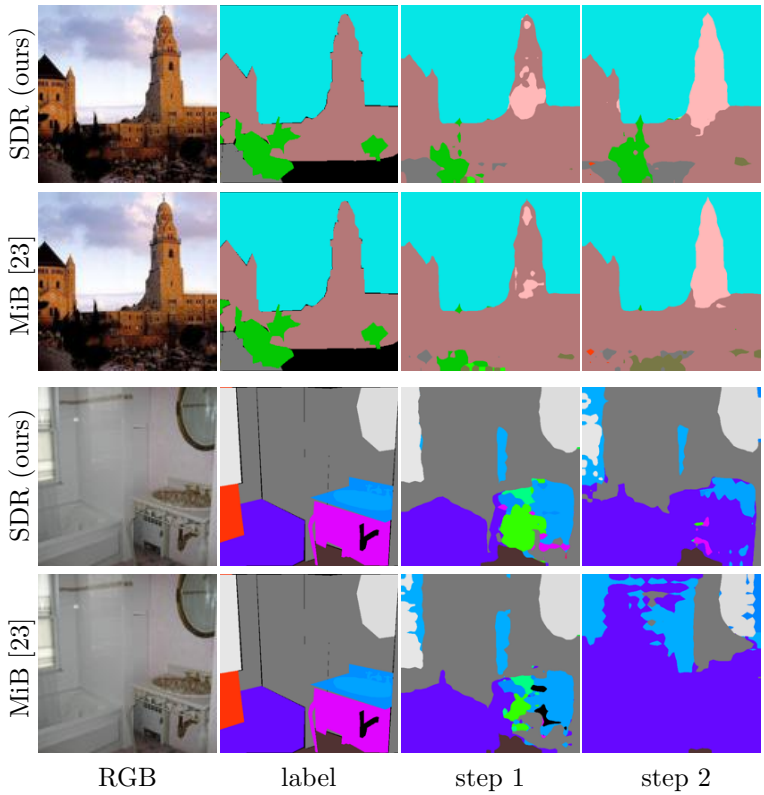


Figure 4.9: Qualitative results on sample scenes in experimental protocol 50-50 on ADE20K during the various incremental steps.

4.5.4 Quantitative Results: per-Class Accuracy

We also analyze the per-class accuracy for all the compared methods in some scenarios. We report the results of per-class IoU and per-class pixel accuracy (PA) on the disjoint 19-1 (Tables 4.3 and 4.4), 15-5 (Tables 4.5 and 4.6) and 15-1 (Tables 4.7 and 4.8) scenarios on the Pascal VOC2012 dataset.

Even when adding as little as 1 class (scenario 19-1 in Tables 4.3 and 4.4) we can appreciate how FT and LwF-MC are generally able to learn the new class to some extent but they catastrophically forget previous classes resulting in a poor final mIoU. This performance drop is typically due to a biased prediction toward the new class (high per-class PA for that class but low IoU). The other competing approaches and our proposal, instead, are more balanced across the various classes and are able to greatly alleviate forgetting (with performance gains distributed across the classes) when learning the new class, thus resulting in higher mIoUs.

Table 4.3: Per-class IoU of compared methods in disjoint experimental protocol on scenario 19-1 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	72.4	62.4	6.7	45.0	47.1	39.5	33.7	40.9	25.7	4.3	54.0	8.0	25.0	50.4	50.6	0.0	35.3	43.0	0.8	59.5	13.2	35.2	13.2	34.2
LwF [49]	87.6	75.4	31.1	71.7	50.8	66.0	81.6	79.0	87.9	32.1	66.9	49.9	84.1	66.2	77.3	79.4	51.8	68.5	42.1	65.8	28.3	65.8	28.3	64.0
LwF-MC [48]	78.6	63.6	0.4	61.2	10.6	35.2	52.8	35.1	75.5	0.4	63.9	1.5	75.5	67.8	32.6	13.1	13.0	63.4	0.7	25.9	1.0	38.5	1.0	36.7
ILT [18]	87.7	79.5	31.6	77.4	54.5	66.5	70.9	79.0	90.4	31.4	66.5	52.9	85.1	67.7	78.1	82.0	56.0	67.3	41.4	72.3	23.4	66.9	23.4	64.8
CIL [55]	85.3	71.4	33.6	75.2	56.5	59.3	45.8	67.2	85.9	27.6	62.7	46.9	85.2	67.9	75.2	83.7	47.4	67.0	42.3	66.0	18.1	62.6	18.1	60.5
MiB [23]	86.9	73.5	35.7	64.0	50.5	71.0	89.5	87.0	84.8	33.7	62.9	56.9	82.1	61.8	79.5	82.4	56.2	62.0	46.0	75.9	26.0	67.0	26.0	65.1
SDR (ours)	89.6	85.3	35.9	78.6	55.2	73.6	86.2	81.9	89.1	34.2	71.4	56.6	86.5	72.7	78.0	83.0	54.1	71.0	45.5	70.4	37.3	69.9	37.3	68.4
SDR+MiB	89.5	84.4	39.0	76.5	53.6	75.1	89.1	87.6	89.0	33.7	67.8	55.4	85.2	72.8	80.8	83.4	57.8	71.3	46.3	78.4	31.4	70.8	31.4	68.9
offline	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	75.5	73.5	75.4

Table 4.4: Per-class pixel accuracy of compared methods in disjoint experimental protocol on scenario 19-1 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	91.5	79.9	7.2	74.9	71.1	44.0	34.3	46.4	26.1	4.5	72.6	8.1	25.4	78.0	53.9	0.0	40.6	58.5	0.8	64.3	82.0	35.2	13.2	34.2
LwF [49]	94.1	85.6	58.7	91.2	59.1	76.3	84.4	80.3	94.1	39.3	93.5	52.3	91.7	95.3	84.0	82.3	76.5	84.1	48.2	68.4	69.6	65.8	28.3	64.0
LwF-MC [48]	99.8	65.5	0.4	63.1	10.7	39.6	53.1	35.3	78.4	0.5	66.5	1.5	77.8	72.0	34.0	13.1	14.4	65.9	0.7	25.9	1.0	38.5	1.0	36.7
ILT [18]	93.5	88.2	59.5	94.3	77.1	83.2	72.0	81.5	96.2	38.7	93.5	55.9	93.8	94.2	84.9	85.7	79.0	91.3	47.1	77.0	63.4	66.9	23.4	64.8
CIL [55]	91.9	77.6	68.6	90.8	66.0	67.6	46.0	67.9	97.3	31.3	95.8	48.6	95.4	94.6	78.9	87.7	82.1	86.4	48.2	68.2	82.1	62.6	18.1	60.5
MiB [23]	89.8	95.0	91.6	97.7	83.9	93.0	93.7	91.2	96.9	52.3	94.2	60.8	96.8	96.2	95.5	88.0	81.9	88.5	56.7	83.6	73.8	67.1	26.1	65.1
SDR (ours)	95.0	90.1	66.5	95.1	67.9	87.7	88.0	83.0	96.4	44.9	93.0	61.3	95.9	95.3	82.7	86.8	81.8	92.9	53.3	72.9	57.9	69.9	37.3	68.4
SDR+MiB	93.1	96.0	86.9	97.3	85.5	91.5	92.1	90.5	96.7	48.8	92.4	58.6	95.7	94.8	91.3	88.9	78.9	90.3	56.1	84.4	69.5	70.8	31.4	68.9
offline	96.1	96.6	85.4	94.4	87.2	92.2	94.7	93.5	96.9	50.2	95.4	56.5	95.8	91.8	94.7	90.8	80.8	92.1	54.8	89.5	83.5	75.5	73.5	75.4

Analyzing the per-class IoUs on the 15-5 case in Tables 4.5 and 4.6 we can appreciate how FT is completely unable to preserve knowledge about previous classes which are heavily forgotten. The competitors can better preserve knowledge related to previous classes while learning new classes but our approach shows superior results in both retaining old classes knowledge and in learning new ones.

The last 15-1 scenario is shown in Tables 4.7 and 4.8. Here we can confirm most of the previous considerations; our method outperforms all the competitors proving its scalability when multiple incremental steps are made. From the per-class pixel accuracy we can observe that most of competing approaches are biased toward the prediction of the very few last classes added to the model, thus reducing the IoU for the other classes.

Table 4.5: Per-class IoU of compared methods in disjoint experimental protocol on scenario 15-5 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	74.2	27.2	0.0	1.6	15.1	11.3	0.0	4.1	0.5	0.0	0.0	0.0	0.0	0.2	0.2	0.0	27.0	25.6	28.9	33.5	52.2	8.4	33.5	14.4
LwF [49]	83.4	59.1	21.7	16.7	36.8	47.0	18.7	62.5	52.3	6.6	4.8	37.7	35.9	44.9	55.5	51.6	22.6	27.8	25.3	39.6	51.1	39.7	33.3	38.2
LwF-MC [48]	85.4	54.2	16.9	59.7	29.7	46.0	34.4	65.9	38.1	5.2	35.9	7.5	62.4	44.3	48.7	29.1	11.4	37.3	8.9	42.1	27.1	41.5	25.4	37.6
ILT [18]	81.7	47.6	18.4	1.6	29.7	19.4	3.8	52.5	56.7	0.5	4.6	20.7	43.1	35.4	33.6	54.8	22.7	22.4	15.9	30.1	34.3	31.5	25.1	30.0
CIL [55]	81.0	45.4	28.8	30.4	31.1	54.5	9.4	67.8	52.1	10.5	9.2	47.9	53.0	35.3	66.3	58.4	23.9	33.3	25.2	39.1	53.9	42.6	35.1	40.8
MiB [23]	78.4	58.3	30.8	52.5	35.5	60.5	60.2	74.8	38.2	14.0	21.6	41.8	42.9	34.8	67.4	48.8	23.2	31.0	24.4	46.3	45.8	47.5	34.1	44.3
SDR (ours)	88.7	82.9	40.5	82.4	62.8	69.2	83.8	88.2	91.6	28.9	71.1	54.2	86.8	80.3	79.7	84.4	39.4	51.4	23.7	63.3	58.7	73.5	47.3	67.2
SDR + MiB	89.4	87.1	39.9	84.8	67.3	75.2	85.1	88.2	91.3	29.9	67.8	54.4	86.1	81.8	80.5	85.0	33.8	43.6	24.7	61.7	56.6	74.6	44.1	67.3
offline	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4

Table 4.6: Per-class pixel accuracy of compared methods in disjoint experimental protocol on scenario 15-5 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	95.3	27.5	0.0	1.6	15.4	11.5	0.0	4.1	0.5	0.0	0.0	0.0	0.0	0.2	0.2	0.0	72.0	90.0	77.2	89.7	80.7	8.4	33.5	14.4
LwF [49]	91.9	79.4	35.4	16.9	50.9	49.0	19.4	71.0	78.8	8.0	5.2	39.7	36.3	78.5	59.2	53.3	67.1	91.2	74.2	81.6	76.5	39.7	33.3	38.2
LwF-MC [48]	96.6	80.7	30.3	68.5	62.0	60.4	37.7	79.7	62.5	10.8	46.2	9.2	73.2	84.4	64.8	31.7	11.4	39.7	9.1	60.1	27.1	41.5	25.4	37.6
ILT [18]	94.6	61.4	26.4	1.6	30.8	19.5	4.0	57.7	71.7	0.5	5.1	20.9	45.9	43.7	34.6	56.7	42.5	86.0	38.8	71.0	44.9	31.5	25.1	30.0
CIL [55]	85.0	80.5	56.3	31.6	57.2	59.5	10.0	81.9	87.6	16.6	12.3	53.9	58.1	86.1	74.1	61.5	84.4	95.7	88.8	93.5	87.1	42.6	35.1	40.8
MiB [23]	80.7	92.6	64.8	64.5	74.0	68.3	65.0	84.3	93.7	23.6	36.2	50.9	49.8	91.2	85.7	52.0	73.9	86.6	87.6	89.9	83.7	47.5	34.1	44.3
SDR (ours)	91.2	95.1	82.1	96.5	80.1	86.3	93.3	92.2	97.0	51.8	93.0	64.3	96.0	91.0	92.0	91.1	68.9	64.1	69.6	74.0	82.9	73.5	47.3	67.2
SDR + MiB	91.7	94.7	80.1	93.4	79.1	88.7	90.6	91.4	96.3	51.0	82.4	64.6	94.9	90.2	91.7	91.8	68.6	67.8	70.3	79.7	81.3	74.6	44.1	67.3
offline	96.1	96.6	85.4	94.4	87.2	92.2	94.7	93.5	96.9	50.2	95.4	56.5	95.8	91.8	94.7	90.8	80.8	92.1	54.8	89.5	83.5	77.5	68.5	75.4

Table 4.7: Per-class IoU of compared methods in disjoint experimental protocol on scenario 15-1 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	70.4	5.5	0.0	5.9	5.2	0.5	0.2	1.6	0.4	0.0	3.3	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.0	9.4	14.8	5.8	4.9	5.6
LwF [49]	77.1	12.0	6.9	52.6	14.3	23.1	18.4	27.3	56.3	20.5	48.9	8.3	17.8	12.6	15.6	8.3	0.0	17.0	21.0	18.6	19.1	26.2	15.1	23.6
LwF-MC [48]	69.5	0.1	0.0	8.0	0.1	7.2	0.0	0.1	8.1	0.0	6.6	0.0	8.0	1.7	0.3	0.1	0.0	0.0	0.0	2.4	8.1	6.9	2.1	5.7
ILT [18]	69.4	0.0	2.1	0.0	0.0	0.1	0.0	4.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	19.2	0.0	0.0	0.0	1.4	4.6	6.7	1.2	5.4
CIL [55]	78.4	2.4	23.6	47.9	4.6	32.9	0.3	29.9	45.4	15.4	30.3	2.4	54.5	13.0	8.7	59.7	15.2	17.5	12.1	20.9	19.2	33.3	15.9	29.1
MiB [23]	70.6	56.2	24.8	41.7	45.8	34.9	44.9	52.8	64.1	17.8	40.4	28.2	16.1	30.3	55.3	0.1	5.9	8.2	16.5	27.2	17.3	39.0	15.0	33.3
SDR (ours)	86.2	47.1	34.2	69.1	37.9	61.3	67.2	72.5	81.1	17.9	51.3	40.8	72.9	67.6	68.5	70.8	8.3	4.8	2.7	24.5	24.2	59.2	12.9	48.1
SDR+MiB	86.9	32.0	29.8	76.0	42.8	60.7	67.4	64.7	85.8	19.2	50.3	39.4	75.1	73.0	69.3	78.2	3.4	2.7	11.5	34.0	20.1	59.4	14.3	48.7
offline	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4

Table 4.8: Per-class pixel accuracy of compared methods in disjoint experimental protocol on scenario 15-1 of Pascal VOC2012.

Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
FT	98.5	5.6	0.0	5.9	5.4	0.5	0.2	1.6	0.4	0.0	3.4	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.0	9.8	80.1	5.8	4.9	5.6
LwF [49]	95.1	12.0	47.8	54.0	15.0	23.2	18.4	27.4	57.3	35.1	62.8	8.3	18.0	12.7	15.7	8.3	0.0	22.0	35.8	47.9	70.7	26.2	15.1	23.6
LwF-MC [48]	99.9	0.1	0.0	8.0	0.1	7.4	0.0	0.1	8.1	0.0	6.6	0.0	8.0	1.7	0.3	0.1	0.0	0.0	0.0	2.9	8.2	6.9	2.1	5.7
ILT [18]	20.9	0.0	73.2	0.0	0.0	0.0	2.3	89.3	19.0	16.3	14.8	1.4	48.3	0.0	23.2	0.4	4.6	0.0	0.0	1.8	4.9	6.7	1.2	5.4
CIL [55]	90.1	16.8	40.0	48.4	15.3	32.7	9.0	28.2	60.1	17.1	75.0	20.4	53.8	28.7	13.5	60.0	31.0	11.8	49.7	50.1	87.0	33.3	15.9	29.1
MiB [23]	72.7	61.7	58.6	60.7	52.3	69.4	45.8	59.2	88.3	30.2	62.3	53.9	68.6	60.7	70.9	0.1	7.0	84.3	28.8	84.9	65.6	39.0	15.0	33.3
SDR (ours)	92.7	47.6	72.3	91.9	44.5	69.2	76.5	74.7	89.3	60.9	92.8	53.1	94.9	75.5	88.3	73.8	11.5	5.1	3.0	35.7	76.6	59.2	12.9	48.1
SDR+MiB	92.7	33.2	45.0	84.7	47.0	67.6	72.1	65.2	96.6	59.1	95.7	45.1	85.3	80.5	83.5	84.2	4.4	2.8	17.2	57.1	76.6	59.4	14.3	48.7
offline	96.1	96.6	85.4	94.4	87.2	92.2	94.7	93.5	96.9	50.2	95.4	56.5	95.8	91.8	94.7	90.8	80.8	92.1	54.8	89.5	83.5	77.5	68.5	75.4

Table 4.9: Ablation on disjoint VOC2012 15-1 in terms of mIoU.

$\mathcal{L}_{ce}$	$\mathcal{L}_{pm}$	$\mathcal{L}_{sp}$	$\mathcal{L}_{cl}$	$\mathcal{L}'_{kd}$	$\mathcal{L}_{kd}$	old	new	all
✓						5.8	4.9	5.6
✓				✓		30.0	11.0	25.4
✓	✓					18.7	9.0	16.4
✓	✓	✓				40.4	12.9	33.9
✓	✓	✓	✓			41.0	13.2	34.4
✓	✓	✓	✓	✓		50.0	**15.9**	41.9
✓	✓	✓	✓		✓	**59.2**	12.9	**48.1**

Table 4.10: Results on standard supervised (non-incremental) semantic segmentation.

$\mathcal{L}_{ce}$	$\mathcal{L}_{sp}$	$\mathcal{L}_{cl}$	$\text{mIoU}_{\text{VOC2012}}$	$\text{mIoU}_{\text{ADE20K}}$
✓			75.4	38.3
✓	✓		75.8	38.7
✓		✓	75.8	38.8
✓	✓	✓	**76.3**	**39.3**

4.6 Ablation Study

To evaluate the effect of each component, we report an ablation analysis in Table 4.9 on the Pascal dataset in the challenging 15-1 scenario. As already noticed, FT leads to a great degradation of mIoU. Early continual semantic segmentation approaches use a classical output-level knowledge distillation [18, 20, 55] which show discrete benefits boosting the mIoU by almost 20%. Each component of the approach significantly contributes to the final mIoU providing non-overlapping and mutual benefits. Matching prototypes, sparsifying features vectors and constraining them via the contrastive objective regularize the latent space bringing large improvements on both old and new classes. We observe that also the contrastive loss brings a significant contribution if applied alone improving the mIoU of 13.5%. Introducing standard output-level knowledge distillation on top increases the accuracy on old classes mainly, and its unbiased version prevents forgetting even more accounting for the background shift across the incremental learning steps.

Finally, we show that two of the proposed approaches (namely, sparsity and contrastive learning) may be beneficial also for the more general case of standard (*i.e.*, non incremental) semantic segmentation. Hence, we conduct some additional experiments on Pascal VOC2012 and ADE20K, reported in Table 4.10, showing the clear benefit of the two components in this setup. On both datasets the outcome is consistent, gaining 0.9% and 1% respectively, even starting from an architecture (*i.e.*, DeepLab-v3+) which is already state of the art.

Random Split. Looking at Table 4.1, we see that in some cases, especially on the 15-1 setup, the proposed method is still far from the offline reference. An interesting question is whether this is due to the difficulty of handling new classes or if, more fundamentally, it is due to an inherent difficulty to train a network using only a small subset of the data at each step. To answer this, we split the dataset equally in 5 parts (each part containing all classes, thus removing the complexity of learning new classes) and then we trained the network sequentially on each of this parts. We obtained 69.9% of mIoU against 75.4% of the joint training, 5.6% of the FT (disjoint) and 48.1% of SDR (disjoint). The difference with respect to joint training is relatively small, and it could be due to sub-optimal network weights estimation (samples are taken from the 5 parts accessed subsequently, instead of the full dataset); on the other side, the

difference with respect to FT is very large proving that handling unseen classes is the key issue and the proposed latent constraints aim at addressing it.

Considerations on Pre-Training. The results reported up to this point have been obtained initializing the weights of the backbone ResNet-101 approach on the ImageNet dataset. This has become the standard setup in continual semantic segmentation approaches [18, 20, 23, 55]. Additional considerations have been already addressed in [20], where it has been shown that pre-training on a segmentation benchmark could boost the accuracy; nonetheless, the ranking of the proposed strategies is mainly maintained.

On the other hand, even ImageNet contains visual samples for many of the elements present in the Pascal dataset (for classification task instead of segmentation), potentially limiting the magnitude of decay on *old* tasks, and likely raising accuracies for *new* concepts that are not necessarily completely new to the encoder. Here, we show how the network performs without such a strong prior on the latent representations. The results are strongly affected by the fact that datasets for in-the-wild segmentation are often too small to reliably train complex deep networks from scratch. We trained on VOC2012 without pre-training and we achieved a low mIoU of 24.4% when training for 30 epochs, and 40.9%, when training for 120 epochs (about 30 hours of computation). In continual learning, the final mIoU are also lower (as the starting point is much lower), but the improvements achieved by our approach and the ranking of the various methods are preserved, for instance in VOC2012 15-1 disjoint the accuracy of SDR (13.5%) is still significantly above FT (4.1%) and MiB (10.9%).

4.7 Design Choices

In this section we present some additional discussion and results motivating the design choices behind the various modules exploited in our work.

Prototypes Matching enforces latent space consistency on old classes, forcing the encoder to produce similar latent representation for previously seen classes in the subsequent steps. The target is achieved by considering the Euclidean distance in the latent space (see Section 4.3.1). Although different distance metrics could have been used in principle (*e.g.*, cosine distance [141, 149, 150]) we found that a simple Euclidean distance was easier to understand and very computationally efficient results similar to more complex schemes.

Contrastive Learning aims at clustering features according to their semantics while tearing apart those of different classes (see Section 4.3.2): we implement it as an attractive force between latent representations with their prototypical representation, against a repulsive one between prototypes of different semantic categories. This attraction-repulsion rule is enforced again using an Euclidean distance metric.

Knowledge Distillation is employed to constraint the decoder to preserve previous knowledge at the output-level and it is implemented as a standard cross-entropy on the output softmax probabilities between old model and current model predictions [18, 20, 23, 55] (see Section 4.3.4).

Sparsity: we think that the most peculiar constraint is represented by the sparsity objective. However, the underlying concept is simple: applying some latent-level sparsification we allow the model to retain enough discriminative power to accommodate the upcoming representations of novel classes without cross-talk with previous ones (see Section 4.3.3). Here, a wide range of possibilities could be considered to address the aforementioned task and one may wonder why the sparsity constraint was designed as it is. First, common sparsity losses are the L0 or L1 norms of feature vectors; however, we show that they achieve lower accuracy. In this work, we define the sparsity objective as the ratio between a stretching function (*i.e.*, the sum of exponentials) and a linear function (*i.e.*, the sum) applied to the feature vectors which were previously normalized

with respect to the maximum value that is assumed by any of the feature channels for that particular class. The idea is that by keeping fixed the sum of features, then the proposed loss in Eq. (4.9) is directly proportional to the degree of distribution across the channels: the value is low when the energy is concentrated in a single or in a few channels, while it increases when distributed (with a gradual change). In some extreme cases, the model of Eq. (4.9) could lead to degenerate solutions, however we argue that these do not happen in practice on a model learning compact representations. We checked to avoid the zero division in the practical implementation, while the *all-ones* case is degenerate in the sense that energy cannot be re-distributed in any way since all channels are already onset to the maximum value and, furthermore, this configuration would not be informative for the decoder.

Although we believe that normalizing the features with a class-conditioned guidance is reasonable (sometimes, features of few particular classes may just be on average more active than features of other classes), we can think of getting rid of it and normalizing with other strategies, *e.g.*, with respect to:

- the maximum value for each feature (*norm max*);
- the overall maximum value (*norm max overall*);
- the *L2* norm of each feature (*norm L2*).

In such cases, Eq. (4.8) would become respectively:

$$\bar{\mathbf{f}}_i = \frac{\mathbf{f}_i}{\max_{f_{i,j} \in \mathbf{f}_i} f_{i,j}} \qquad \mathbf{f}_i \in \mathbf{F}_n \tag{4.10}$$

$$\bar{\mathbf{f}}_i = \frac{\mathbf{f}_i}{\max_{g_{j,l} \in \mathbf{g}_j} g_{j,l}} \qquad \mathbf{f}_i, \mathbf{g}_j \in \mathbf{F}_n \tag{4.11}$$

$$\bar{\mathbf{f}}_i = \frac{\mathbf{f}_i}{||\mathbf{f}_i||_2} \qquad \mathbf{f}_i \in \mathbf{F}_n \tag{4.12}$$

Furthermore, in principle any stretching function could be applied in spite of the sum of exponentials over the linear sum. For instance, the sum of squares (*power 2*) or sum of the cubic powers (*power 3*) could be used as stretching functions: *i.e.*, formulating Eq. (4.9) respectively as:

$$\mathcal{L}_{sp} = \frac{1}{|\mathbf{f}_i \in \mathbf{F}_n|} \sum_{\mathbf{f}_i \in \mathbf{F}_n} \frac{\sum_j \bar{f}_{i,j}^2}{\sum_j \bar{f}_{i,j}} \tag{4.13}$$

$$\mathcal{L}_{sp} = \frac{1}{|\mathbf{f}_i \in \mathbf{F}_n|} \sum_{\mathbf{f}_i \in \mathbf{F}_n} \frac{\sum_j \bar{f}_{i,j}^3}{\sum_j \bar{f}_{i,j}}. \tag{4.14}$$

Finally, following the success of recent works exploiting entropy minimization [151] techniques, an alternative strategy could be to minimize the entropy of the latent representations opportunely preceded by *L1* or *softmax* normalization of each feature vector in order to obtain a probability distribution over the channels. More formally:

$$\bar{\mathbf{f}}_i = \frac{\mathbf{f}_i}{||\mathbf{f}_i||_1} \qquad \mathbf{f}_i \in \mathbf{F}_n \tag{4.15}$$

$$\bar{\mathbf{f}}_i = \frac{\exp(\mathbf{f}_i)}{\sum_j \exp(f_{i,j})} \qquad \mathbf{f}_i \in \mathbf{F}_n \tag{4.16}$$

$$\mathcal{L}_{sp} = \frac{1}{|\mathbf{f}_i \in \mathbf{F}_n|} \sum_{\mathbf{f}_i \in \mathbf{F}_n} \sum_j -\bar{f}_{i,j} \cdot \log\left(\bar{f}_{i,j}\right) \tag{4.17}$$

Table 4.11 shows the performance of the aforementioned approaches in the 19-1 and 15-1 disjoint scenarios on Pascal VOC2012. Different normalization rules bring to consistently lower results, proving the efficacy of using class guidance during normalization. Also, different stretching functions are found to be less adequate for our purpose reducing the final mIoU of about 2% to 4%. Finally, entropy minimization techniques obtain competitive and comparable results in the $15-1$ scenario, while they experience a drop of about $2-3\%$ of mIoU when only one class is added.

Table 4.11: Comparison of different $\mathcal{L}_{sp}$ in terms of mIoU in the disjoint scenarios 19-1 and 15-1 on Pascal VOC2012 dataset.

Method	mIoU_{19-1}	mIoU_{15-1}
L0	66.7	46.3
L1	65.9	45.4
norm max	67.4	47.8
norm max overall	67.5	45.6
norm L2	64.8	44.3
power 2	66.3	44.2
power 3	66.6	45.3
entropy (L1)	65.3	48.0
entropy (softmax)	66.0	48.0
ours	**68.4**	**48.1**

4.8 Summary

In this section, we presented some latent representation shaping techniques to prevent forgetting in continual semantic segmentation. In particular, the proposed constraints on the latent space regularize the learning process reducing forgetting whilst simultaneously improving the recognition of novel classes. A prototypes matching constraint enforces latent space consistency on old classes, a features sparsification objective reduces the number of active channels limiting cross-talk between features of different classes, and contrastive learning clusters features according to their semantic while tearing apart those of different classes. Our evaluation shows the effectiveness of the proposed techniques, which can also be seamlessly applied in combination of previous methods (*e.g.*, knowledge distillation).

To further boost accuracy and reduce forgetting of previous categories, in the next chapter we describe how we can incorporate GANs or web-crawled data inside our framework to reproduce images containing previous classes.

5 Replay-based Continual Learning in Semantic Segmentation

5.1 Introduction

This chapter investigates the usage of additional replay data containing previously learned classes in order to reduce forgetting of such categories.

As we have already observed, current approaches for class-incremental semantic segmentation re-frame knowledge distillation strategies inspired by previous works on image classification [18, 20, 23, 55]. Although they partially alleviate forgetting, they often fail when multiple incremental steps are performed or when background shift [23] (*i.e.*, change of statistics of the background across learning steps, as it incorporates old or future classes) occurs.

In this chapter, we present a completely different strategy and, instead of distilling knowledge from a teacher model (*i.e.*, the old one) to avoid forgetting, we propose to re-create samples of old classes by using replay strategies. We propose RECALL (*REplay in ContinuAL Learning*), a method that re-generates representations of old classes and mixes them with the available training data, *i.e.*, containing novel classes being learned (see Figure 5.1). To reduce background shift we introduce a self-inpainting strategy that re-assigns the background region according to predictions of the previous model. To generate representations of past classes we pursue two possible directions. The first is based on a pre-trained generative model, *i.e.*, a Generative Adversarial Network (GAN) [108] conditioned to produce samples of the required input classes. The GAN has been trained beforehand on a dataset different than the target one (we chose ImageNet as it comprehends a wide variety of classes and domains), thus requiring a Class Mapping Module to perform the translation between the two label spaces. The second strategy, instead, is based on crawling images from the web, querying the class names to drive the search. Both approaches allow to retrieve a large amount of weakly labeled data. Finally, we generate pseudo-labels for semantic segmentation using a side labeling module, which requires only minimal extra storage.

The methodologies presented in this chapter can be seamlessly applied on top of other existing techniques already proposed at the output space and at the feature level.

5.1.1 Preliminaries

As we have observed in Sections 2.3 and 2.5.4, generative replay approaches [71, 73, 110] rely on generative models typically trained on the same data distribution, which are later used to

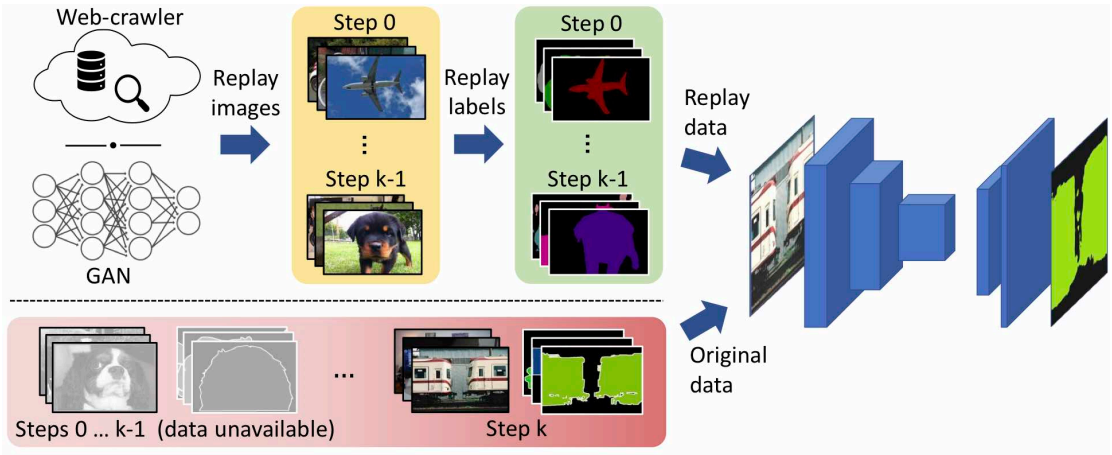


Figure 5.1: Replay images of previously seen classes are retrieved by a web crawler or a generative network and further labeled. Then, the network is incrementally trained with a mixture of new and replay data.

generate artificial samples to preserve previous knowledge.

Generative models are usually GANs [71, 73, 153] or auto-encoders [110]. In this chapter we employ two kinds of generative replays: either resorting to a standard pre-trained GAN or to web-crawled images to avoid forgetting, without storing any of the samples related to previous tasks. When using the generative model, differently from previous works on continual image classification, we do not select real exemplars as anchor to support the learned distribution [153] nor we train or fine-tune the GAN architecture on the current data distribution [71, 73, 153], thus reducing memory and computation time.

Webly-supervised learning is an emerging paradigm in which large amounts of web data is exploited for learning CNNs [154–156]. Recently, it was employed also in semantic segmentation to provide a plentiful source of both images [157,158] and videos [159] with weak image-level class labels during training. The most active research directions are devoted toward understanding how to query images, how to filter and exploit them (*e.g.*, assigning pseudo-labels). To the best of our knowledge, however, webly-supervised learning has not yet been explored in continual learning as a replay strategy.

5.1.2 Contributions

Our main contributions are:

1. we propose RECALL, which is the first approach to use replay data in continual semantic segmentation;

2. to the best of our knowledge, we are the first to introduce the webly-supervised paradigm in continual learning, showing how we can extract useful clues from extremely weakly supervised and noisy samples;

3. we devise a background inpainting strategy to generate pseudo-labels and overcome the background shift;

4. we achieve state-of-the-art results on a wide range of scenarios, especially when performing multiple incremental steps.

5.2 Problem Formulation and Setup

In semantic segmentation, given an image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$, we aim at producing a map $\hat{\mathbf{Y}} \in \mathcal{Y} \subset \mathcal{C}^{H \times W}$ that is a prediction of the ground truth map $\mathbf{Y}$. This is nowadays usually achieved by using a suitable deep learning model $M : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}|}$, commonly made by a feature extractor E followed by a decoding module D, *i.e.*, $M = D \circ E$.

In standard supervised learning, the model is learned in a single shot over a training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$, available in its complete form to the training algorithm. In class-incremental learning, instead, we assume that the training is performed in multiple steps and only a subset of training data is made available to the algorithm at each step $k = 0, ..., K$. More in detail, we start from an initial step $k = 0$ where only training data concerning a subset of all the classes $\mathcal{C}_0 \subset \mathcal{C}$ is available (we assume that the special background class $b \in \mathcal{C}_0$). We denote with $M_0 : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}_0|}$, $M_0 = D_0 \circ E_0$ the model trained after this initial step. Moving to a generic step k, a new set of classes $\mathcal{C}_k$ is added to the class collection $\mathcal{C}_{0 \to (k-1)}$ learned up to that point, resulting in an expanded set of learnable classes $\mathcal{C}_{0 \to k} = \mathcal{C}_{0 \to (k-1)} \cup \mathcal{C}_k$ (we assume $\mathcal{C}_{0 \to (k-1)} \cap \mathcal{C}_k = \varnothing$). The model after the k-th step of training is $M_k : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}_{0 \to k}|}$, where $M_k = D_k \circ E_0$, since in our approach the encoder E_0 is not trained during the incremental steps and only the decoder is updated [18].

Two main continual scenarios have been proposed (*i.e.*, disjoint and overlapped, see Section 2.4 and related works [18, 21, 23] for a more detailed description) and we tackle both in a unified framework.

Disjoint setup: in the initial step all the images in the training set with at least one pixel belonging to a class of $\mathcal{C}_0$ (except for b) are assumed to be available. We denote with $\mathcal{Y}_{\mathcal{C}_0 \cup \{b\}} \subset \mathcal{C}_0^{H \times W}$ the corresponding output space where labels can only belong to $\mathcal{C}_0$, while all the pixels not pertaining to these classes are assigned to b. The incremental partitions are built as *disjoint* subsets of the whole training set. The training data associated to k-th step, $\mathcal{T}_k \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_k \cup \{b\}}$, contains only images corresponding to classes in $\mathcal{C}_k$ with just classes of step k annotated (old classes are labeled as b), and is disjoint with respect to previous and past partitions.

Overlapped setup: in the first phase we select the subset of training images having only $\mathcal{C}_0$-labeled pixels. Then, the training set at each incremental step contains *all* the images with labeled pixels from $\mathcal{C}_k$ *i.e.* $\mathcal{T}_k \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_k \cup \{b\}}$. Similarly to the initial step, labels are limited to semantic classes in $\mathcal{C}_k$, while remaining pixels are assigned to b.

In both setups, b undergoes a semantic shift at each step, as pixels of ever changing class sets are assigned to it.

5.3 General Architecture

In the standard setup, the segmentation model M is trained with annotated samples from a training set $\mathcal{T}$. Data should be representative of the task we would like to solve, meaning that multiple instances of all the considered semantic classes $\mathcal{C}$ should be available in the provided dataset for the segmentation network to properly learn them. Once $\mathcal{T}$ has been assembled, the cross-entropy objective is commonly employed to optimize the weights of M:

$$\mathcal{L}_{ce}(M; \mathcal{C}, \mathcal{T}) = -\frac{1}{|\mathcal{T}|} \sum_{\mathbf{X}, \mathbf{Y} \in \mathcal{T}} \sum_{c \in \mathcal{C}} \mathbf{Y}[c] \cdot \log \big(M(\mathbf{X})[c] \big) \tag{5.1}$$

In the incremental learning setting, when performing an incremental training step k only samples related to new classes $\mathcal{C}_k$ are assumed to be at our disposal. Following the simplest approach,

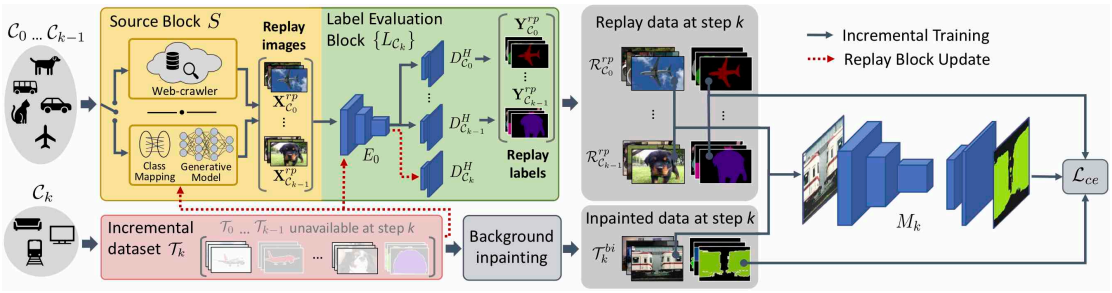


Figure 5.2: Overview of the proposed RECALL: class labels from past incremental steps are provided to a Source Block, either a web crawler or a pre-trained conditional GAN, which retrieves a set of unlabeled replay images for the past semantic classes. Then, a Label Evaluation Block produces the missing annotations. Finally, the segmentation network is incrementally trained with a replay-augmented dataset, composed of both new classes data and replay data.

we could initialize our model's weights from the previous step (M_{k-1}, $k \geq 1$) and learn the segmentation task over classes $\mathcal{C}_{0\to k}$ by optimizing the standard objective $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\to k}, \mathcal{T}_k)$ with data from the current training partition $\mathcal{T}_k$. However, simple fine-tuning leads to catastrophic forgetting, being unable to preserve previous knowledge. Our framework is depicted in Figure 5.2.

Architecture of Replay Block. To cope with this issue, we opt for a replay strategy. Our goal is to retrieve task-related knowledge of past classes to be blended into the ongoing incremental step, all without accessing training data of previous iterations. To this end, we introduce a Replay Block, whose target is twofold. First, it has to provide images resembling instances of classes from previous steps, whether generating them from scratch or retrieving them from an available alternative source (*e.g.*, a web database). Second, it has to obtain reliable semantic labels of those images, by resorting to learned knowledge from past steps.

The Replay Block's image retrieval task is executed by what we call Source Block:

$$S : \mathcal{C}_k \mapsto \mathcal{X}^{rp}_{\mathcal{C}_k} \tag{5.2}$$

This module takes in input a set of classes $\mathcal{C}_k$ (background excluded) and provides images whose semantic content can be ascribed to those categories (*e.g.*, $\mathbf{X}^{rp} \in \mathcal{X}^{rp}_{\mathcal{C}_k}$). We adopt two different solutions for the Source Block, namely GAN and web-based techniques, both detailed in Section 5.4.

The Source Block provides unlabeled image data (if we exclude the weak image-level classification labels), and for this reason we introduce an additional Label Evaluation Block $\{L_{\mathcal{C}_k}\}_{\mathcal{C}_k \subset \mathcal{C}}$, which aims at annotating examples provided by the replay module. This block is made of separate instances $L_{\mathcal{C}_k} = D^H_{\mathcal{C}_k} \circ E_0$, each denoting a segmentation model to classify a specific set of semantic categories $\mathcal{C}_k \cup \{b\}$ (*i.e.*, the classes in $\mathcal{C}_k$ plus the background) :

$$L_{\mathcal{C}_k} : \mathcal{X}_{\mathcal{C}_k} \mapsto \mathbb{R}^{H \times W \times (|\mathcal{C}_k \cup \{b\}|)} \tag{5.3}$$

All $L_{\mathcal{C}_k}$ modules share the encoder section E_0 from the initial training step, so that only a minimal portion of the segmentation network (*i.e.*, $D^H_{\mathcal{C}_k}$, which accounts for only few parameters, see Section 5.6.3) is stored for each block's instance. Notice that a single instance recognizing all classes could be used, leading to an even more compact representation, but it experimentally led to worse performance.

Provided that S and $L_{\mathcal{C}_k}$ are available, replay training data can be collected for classes in $\mathcal{C}_k$. A query to S outputs a generic image example $\mathbf{X}^{rp}_{\mathcal{C}_k} = S(\mathcal{C}_k)$, which is then associated to

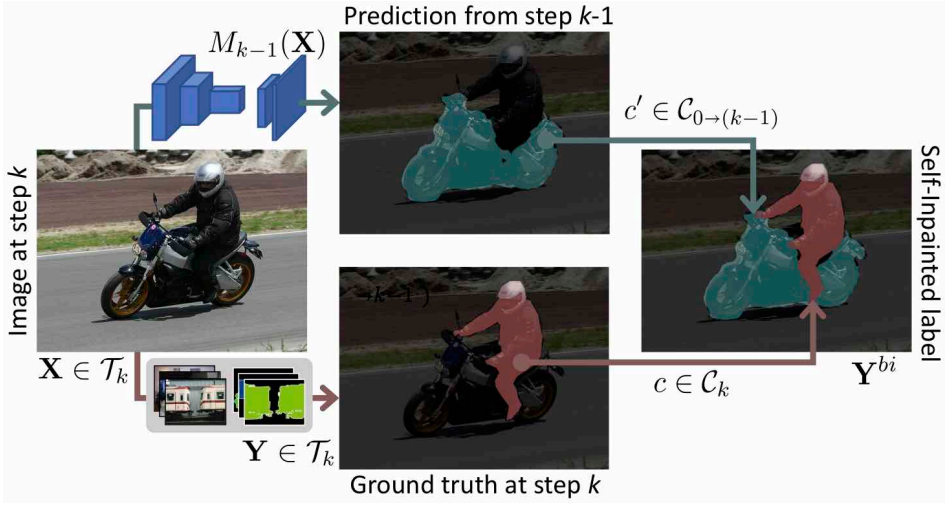


Figure 5.3: Background self-inpainting process.

its prediction $\mathbf{Y}^{rp}_{\mathcal{C}_k} = \underset{c\in\mathcal{C}_k\cup\{b\}}{\arg\max}\, L_{\mathcal{C}_k}(\mathbf{X}^{rp}_{\mathcal{C}_k})[c]$. By retrieving multiple replay examples, we build a replay dataset $\mathcal{R}_{\mathcal{C}_k} = \{(\mathbf{X}^{rp}_{\mathcal{C}_k}, \mathbf{Y}^{rp}_{\mathcal{C}_k})_n\}_{n=1}^{N_r}$, where N_r is a fixed hyperparameter empirically set (see Section 5.5).

Background Self-Inpainting. To deal with the background shift phenomenon, we propose a simple yet effective inpainting mechanism to transfer knowledge from the previous model into the current one. While the replay block re-creates samples of previously seen classes, background inpainting acts on background regions of current samples reducing the background shift and at the same time bringing a regularization effect similar to knowledge distillation [18, 23], although its implementation is quite different. At every step k with training set $\mathcal{T}_k$, we take the background region of each ground truth map and we label it with the associated prediction from the previous model M_{k-1} (see Figure 5.3). We call it *background inpainting* since the background regions in label maps are changed according to a self-teaching scheme based on the prediction of the old model. More formally, we replace each original label map $\mathbf{Y}$ available at step $k > 0$ with its inpainted version $\mathbf{Y}^{bi}$:

$$\mathbf{Y}^{bi}[h,w] = \begin{cases} \mathbf{Y}[h,w] & \text{if } \mathbf{Y}[h,w] \in \mathcal{C}_k \\ \underset{c\in\mathcal{C}_{0\to k-1}}{\arg\max}\, M_{k-1}(\mathbf{X})[h,w][c] & \text{otherwise} \end{cases} \tag{5.4}$$

where $(\mathbf{X}, \mathbf{Y}) \in \mathcal{T}_k$, while $[h, w]$ denotes the pixel coordinates. Labels at step $k = 0$ are not inpainted, as at that stage we lack any prior knowledge of past classes. When background inpainting is performed, each set $\mathcal{T}^{bi}_k \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_{0\to k}}$ $(k > 0)$ contains all samples of $\mathcal{T}_k$ after being inpainted.

Incremental Training with Replay Block. The training procedure of RECALL is detailed and summarized in Algorithm 5.1 and the process is described in Figure 5.2. Suppose we are at the incremental step k, with only training data of classes in $\mathcal{C}_k$ from partition $\mathcal{T}_k$ available. In a first stage, the Replay Block is fixed and it is used to retrieve annotated data for steps from 0 to $k - 1$ uniformly distributed among all the past classes. Following the described pipeline, the generative and labeling models are applied independently over each incremental class set

Algorithm 5.1 RECALL: incremental training procedure.

Input: $\{\mathcal{T}_k\}_{k=0}^{K}$ and $\{\mathcal{C}_k\}_{k=0}^{K}$
Output: M_K
train $M_0 = E_0 \circ D_0$ with $\mathcal{L}_{ce}(M_0; \mathcal{C}_0, \mathcal{T}_0)$
train S on $(\mathcal{C}_0, \mathcal{T}_0)$
train $D_{\mathcal{C}_0}^{H}$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_0}; \mathcal{C}_0, \mathcal{T}_0)$
for $k \leftarrow 1$ to K **do**
  background inpainting on $\mathcal{T}_k$ to obtain $\mathcal{T}_k^{bi}$
  train S on $(\mathcal{C}_k, \mathcal{T}_k)$
  train $D_{\mathcal{C}_k}^{H}$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_k}; \mathcal{C}_k \cup \{b\}, \mathcal{T}_k)$
  generate $\mathcal{T}_k^{rp} = \mathcal{T}_k^{bi} \cup \mathcal{R}_{\mathcal{C}_{0\rightarrow(k-1)}}$
  train D_k with $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\rightarrow k}, \mathcal{T}_k^{rp})$
end for

$\mathcal{C}_i, i = 0, ..., k-1$. The replay training dataset for step k is the union of the single replay sets for each previous step: $\mathcal{R}_{\mathcal{C}_{0\rightarrow(k-1)}} = \bigcup_{i=0}^{k-1} \mathcal{R}_{\mathcal{C}_i}$. Once we have assembled $\mathcal{R}_{\mathcal{C}_{0\rightarrow(k-1)}}$, by merging it with $\mathcal{T}_k^{bi}$ we get an augmented step-k training partition $\mathcal{T}_k^{rp} = \mathcal{T}_k^{bi} \cup \mathcal{R}_{\mathcal{C}_{0\rightarrow(k-1)}}$. This new set, in principle, is complete of annotated samples containing both old and new classes, thanks to replay data. Therefore, we effectively learn the segmentation model M_k through the cross-entropy objective $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\rightarrow k}, \mathcal{T}_k^{rp})$ on replay-augmented training data. This mitigates the bias toward new classes, thus preventing forgetting.

In a second stage, we exploit $\mathcal{T}_k$ to train the Class Mapping Module if needed (see Section 5.4). In particular, we teach the Source Block S to produce samples of $\mathcal{C}_k$, and we optimize the decoder $D_{\mathcal{C}_k}^{H}$ to correctly segment, in conjunction with E_0, images from $\mathcal{T}_k$ by minimizing $\mathcal{L}_{ce}(L_{\mathcal{C}_k}; \mathcal{C}_k \cup \{b\}, \mathcal{T}_k)$. This stage is not exploited in the current step, but will be necessary in future ones.

During a standard incremental training stage, we follow a mini-batch gradient descent scheme, where batches of annotated training data are sampled from $\mathcal{T}_k^{rp}$. However, to guarantee a proper stream of information, we opt for an interleaving sampling policy, rather than a random one. In particular, at a generic iteration of training, a batch of data $\mathcal{B}^{rp}$ supplied to the network is made of r_{new} samples from the current training partition $\mathcal{T}_k^{bi}$ and r_{old} replay samples from $\mathcal{R}_{\mathcal{C}_{0\rightarrow(k-1)}}$. The ratio between r_{new} and r_{old} controls the proportion of replay and new data (see also Section 5.6.3). We need, in fact, to carefully balance how new data is dosed with respect to replay one, so that enough information about new classes is provided within the learning process, while concurrently we assist the network in recalling knowledge acquired in past steps to prevent catastrophic forgetting.

5.4 Replay Strategies

In this section we describe more in detail the replay strategies employed for the image generation task of the Source Block S. As mentioned previously, we opt for a generative approach based on a GAN framework and for an online retrieval solution, where images are collected by a web crawler.

Replay by GAN. The GAN-based strategy exploits a deep generative adversarial framework to re-create the no longer available samples for previously seen classes. We use a conditional GAN, G, pre-trained on a generic large-scale visual dataset with data from a wide set of semantic

classes $\mathcal{C}^G$ and different domains. For the experiments, we choose an ImageNet [121] based pre-training. On this regard, we remark that classes and domains are not required to be completely coherent: for instance *person* does not exist in ImageNet, but related classes (*e.g.*, *hat*) still allow to preserve its knowledge (further considerations on this are reported in Section 5.6.5). When performing the k-th incremental step, we retrieve images containing previously seen classes by sampling the GAN's generator output, *i.e.*, $\mathbf{X}^{rp} = G(\mathbf{n}, c^G)$ conditioned on GAN's classes $c^G \in \mathcal{C}^G$ corresponding to the target ones from the original training data ($\mathbf{n}$ is a generic noise input).

Since the GAN is pre-trained on a separate dataset, typically it inherits a different label set. For this reason, the Source Block with GAN is composed of two main modules, namely the actual GAN for image generation and a Class Mapping Module to translate each class of the semantic segmentation incremental dataset to the most similar class of the GAN's training dataset. Provided that we have trained both the GAN and class mapping modules, first we use the latter to translate the class set $\mathcal{C}_k$ to the matching set $\mathcal{C}_k^G$. Then, a set of queries to the conditioned GAN's generator:

$$\mathbf{X}_{\mathcal{C}_k}^{rp} = G(\mathbf{n}, c^G), \quad c^G \in \mathcal{C}_k^G \tag{5.5}$$

provides samples resembling the ones in $\mathcal{C}_k$, as long as the mapping is able to properly associate each original class to a statistically similar counterpart in the GAN's label space.

At each incremental step k, the Source Block with GAN goes through two separate training and inference stages. In a first training phase, samples from $\mathcal{T}_k$ are fed to an Image Classifier I, which is pre-trained to solve an image classification task on the GAN's dataset. In particular, for each class $c \in \mathcal{C}_k$ we select the corresponding training subset $\mathcal{T}_k^c \subset \mathcal{T}_k$, *i.e.*, all the samples of set $\mathcal{T}_k$ associated to class c, and we sum the resulting class probability vectors from the classification output. Then, the GAN's class c^G with the highest probability score is identified by:

$$c^G = \arg\max_{j \in \mathcal{C}^G} \sum_{\mathbf{X} \leftarrow \mathcal{T}_k^c} I(\mathbf{X})\,[j] \tag{5.6}$$

where $\mathbf{X}$ is extracted from $\mathcal{T}_k^c$ (labels are not used) and $I(\mathbf{X})$ denotes the vector output of the last softmax layer of I, whose j-th entry corresponds to the j-th GAN's class. By repeating this procedure for every class in $\mathcal{C}_k$, we build the mapped set $\mathcal{C}_k^G$. Class correspondence is stored, so that at each step we have access to class mappings of past iterations.

In a second evaluation phase, classes in $\mathcal{C}_{0\rightarrow(k-1)}$ are given as input to the Source Block. Thanks to the class correspondences saved in previous steps, $\mathcal{C}_{0\rightarrow(k-1)}$ are mapped to $\mathcal{C}_{0\rightarrow(k-1)}^G$. Next, image generation conditioned on each class of $\mathcal{C}_{0\rightarrow(k-1)}^{GAN}$ is performed, and the resulting replay images are fed to the Label Evaluation Block to be associated to their corresponding semantic labels. By following this procedure, we end up with self-annotated data of past classes suitable to support the supervised training at the current step, which otherwise would be limited to new classes.

Replay by Web Crawler. As an alternative we propose to retrieve training examples from an online source. For the evaluation, we searched images from the *Flickr* website, but any other online database or search engine can be used.

Assuming we are at the incremental step k and we have access to the names of every class in the past iterations (*e.g.*, $\forall c \in \mathcal{C}_{0\rightarrow(k-1)}$), we download images whose tag and description happen to both contain the class name through the *Flickr*'s web crawler. Then, the web-crawled images are fed to the Label Evaluation Block for their annotation.

Compared to the GAN-based approach, the online retrieval solution is simpler as no learnable

modules are introduced. In addition, we completely avoid to assume that a larger dataset is available, whose class range should be sufficiently ample and diverse to cope with the continuous stream of novel classes incrementally introduced. On the other side, this approach requires the availability of an internet connection and in some way exploits additional training data even if almost unsupervised. Plus, we lack control over the weak labeling performed by the web source.

5.5 Implementation Details

We use the DeepLab-V2 [5] as segmentation architecture with ResNet-101 [152] as backbone. Nonetheless, RECALL is independent of the specific network architecture. Encoder's weights are pre-trained on ImageNet [121] and all network's weights are trained in the initial step 0. In the following steps, only the main decoder is trained, together with the additional $\{D^H_{\mathcal{C}_k}\}_k$ helper decoders, which are needed to annotate replay samples (as discussed in Section 5.3). For a fair comparison, all competing approaches are trained with the same backbone. SGD with momentum is used for weights optimization, with initial learning rate set to 5×10^{-4} and decreased to 5×10^{-6} according to polynomial decay of power 0.9. Following previous works [18, 20], we train the model for $|\mathcal{C}_k| \times 1000$ learning steps in the disjoint setup and for $|\mathcal{C}_k| \times 1500$ steps in the overlapped setup. Each helper decoder $D^H_{C_k}$ is trained with a polynomially decaying learning rate starting from 2×10^{-4} and ending at 2×10^{-6} for $|\mathcal{C}_k| \times 1000$ steps. As Source Block, we use BigGAN-deep [160] pre-trained [161] on ImageNet. At each incremental step k, we generate 500 replay samples per old class, *i.e.* $N_r = 500$. To map classes from the segmentation dataset to the GAN's one, we use the EfficientNet-B2 [162] classifier implemented at [163] and pre-trained on ImageNet. The interleaving ratio r_{old}/r_{new} is set to 1.

As input pre-processing, random scaling and mirroring are followed by random padding and cropping to 321×321 px. The entire framework is developed in TensorFlow [125] and trained on a single NVIDIA RTX 2070 Super. Training time varies depending on the setup, with the longest run taking about 5 hours. Code and replay data are available at `https://github.com/LTTM/RECALL`.

5.6 Experimental Results

5.6.1 Analyses on Pascal VOC2012

In this section we present the experimental evaluation on the Pascal VOC2012 dataset [114]. Following previous works on this topic [18, 20, 23, 51], we start by analyzing the performance on three widely used incremental scenarios: *i.e.*, addition of the last class (19-1), addition of the last 5 classes at once (15-5) and addition of the last 5 classes sequentially (15-1). Moreover, we report the performance on three more challenging scenarios in which 10 classes are added sequentially one by one (10-1), in 2 batches of 5 elements each (10-5) and all at once (10-10). Classes for the incremental steps are selected according to the alphabetical order. We compare with the naïve fine-tuning approach (FT), which defines the lower limit to the accuracy of an incremental model, and with the joint training on the complete dataset in one step, which serves as upper bound. We also report the results of a simple Store and Replay (S&R) method, where at each incremental step we store a certain number of true samples for newly added classes, such that the respective size in average matches the size of the helper decoders needed by RECALL (see Figure 5.8). As comparison, we include 2 methods extended from classification (*i.e.*, LwF [49] and its single-headed version LwF-MC [48]) and the most relevant methods designed for continual segmentation (*i.e.*, ILT [18], CIL [55], MiB [23] and SDR [21]). Exhaustive quantitative results

Table 5.1: mIoU on Pascal VOC2012 for different incremental setups. Results of competitors in the upper part come from [21, 23], while we run their implementations for the new scenarios in the bottom part.

	19-1						15-5						15-1					
	Disjoint			Overlapped			Disjoint			Overlapped			Disjoint			Overlapped		
Method	1-19	20	all	1-19	20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all
FT	35.2	13.2	34.2	34.7	14.9	33.8	8.4	33.5	14.4	12.5	36.9	18.3	5.8	4.9	5.6	4.9	3.2	4.5
S&R	55.3	43.2	56.2	54.0	48.0	55.1	38.5	43.1	41.6	36.3	44.2	40.3	41.0	31.8	40.7	38.6	31.2	38.9
LwF [49]	65.8	28.3	64.0	62.6	23.4	60.8	39.7	33.3	38.2	67.0	41.8	61.0	26.2	15.1	23.6	24.0	15.0	21.9
LwF-MC [48]	38.5	1.0	36.7	37.1	2.3	35.4	41.5	25.4	37.6	59.8	22.6	51.0	6.9	2.1	5.7	6.9	2.3	5.8
ILT [18]	66.9	23.4	64.8	50.2	29.2	49.2	31.5	25.1	30.0	69.0	46.4	63.6	6.7	1.2	5.4	5.7	1.0	4.6
CIL [55]	62.6	18.1	60.5	35.1	13.8	34.0	42.6	35.0	40.8	14.9	37.3	20.2	33,3	15.9	29.1	6.3	4.5	5.9
MiB [23]	69.6	25.6	67.4	**70.2**	22.1	67.8	71.8	43.3	64.7	**75.5**	49.4	69.0	46.2	12.9	37.9	35.1	13.5	29.7
SDR [21]	**69.9**	37.3	**68.4**	69.1	32.6	67.4	**73.5**	47.3	**67.2**	75.4	52.6	**69.9**	59.2	12.9	48.1	44.7	21.8	39.2
RECALL (GAN)	65.2	**50.1**	65.8	67.9	53.5	68.4	66.3	49.8	63.5	66.6	50.9	64.0	66.0	44.9	62.1	65.7	47.8	62.7
RECALL (Web)	65.0	47.1	65.4	68.1	**55.3**	**68.6**	69.2	**52.9**	66.3	67.7	**54.3**	65.6	**67.6**	**49.2**	**64.3**	**67.8**	**50.9**	**64.8**
Joint	75.5	73.5	75.4	75.5	73.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4

	10-10						10-5						10-1					
	Disjoint			Overlapped			Disjoint			Overlapped			Disjoint			Overlapped		
Method	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all
FT	7.7	60.8	33.0	7.8	58.9	32.1	7.2	41.9	23.7	7.4	37.5	21.7	6.3	2.0	4.3	6.3	2.8	4.7
S&R	25.1	53.9	41.7	18.4	53.3	38.2	26.0	28.5	29.7	22.2	28.5	27.9	30.2	19.3	27.3	28.3	20.8	27.1
LwF [49]	63.1	61.1	62.2	**70.7**	63.4	**67.2**	52.7	47.9	50.4	55.5	47.6	51.7	6.7	6.5	6.6	16.6	14.9	15.8
LwF-MC [48]	52.4	42.5	47.7	53.9	43.0	48.7	44.6	43.0	43.8	44.3	42.0	43.2	6.9	1.7	4.4	11.2	2.5	7.1
ILT [18]	**67.7**	**61.3**	**64.7**	70.3	61.9	66.3	53.4	48.1	50.9	55.0	44.8	51.7	14.1	0.6	7.5	16.5	1.0	9.1
CIL [55]	37.4	60.6	48.4	38.4	60.0	48.7	27.5	41.4	34.1	28.8	41.7	34.9	7.1	2.4	4.9	6.3	0.8	3.6
MiB [23]	66.9	57.5	62.4	70.4	63.7	67.2	54.3	47.6	51.1	55.2	49.9	52.7	14.9	9.5	12.3	15.1	14.8	15.0
SDR [21]	67.5	57.9	62.9	70.5	**63.9**	**67.4**	55.5	48.2	52.0	56.9	51.3	54.2	25.5	15.7	20.8	26.3	19.7	23.2
RECALL (GAN)	62.6	56.1	60.8	65.0	58.4	63.1	60.0	52.5	57.8	60.8	52.9	58.4	58.3	46.0	53.9	59.5	46.7	54.8
RECALL (Web)	64.1	56.9	61.9	66.0	58.8	63.7	**63.2**	**55.1**	**60.6**	**64.8**	**57.0**	**62.3**	**62.3**	**50.0**	**57.8**	**65.0**	**53.7**	**60.7**
Joint	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4

in terms of mIoU are shown in Table 5.1. For each setup we report the mean accuracy for the initial set of classes, for the classes in the incremental steps and for all classes, computed after the overall training.

Addition of the last class. First, we train over the first 19 classes during step 0. Then, we perform a single incremental step to learn *tv/monitor*. Looking at Table 5.1 (upper-left part), we notice that FT results in a drastic performance degradation with respect to joint training, due to catastrophic forgetting. RECALL, instead, shows higher overall mIoU than competitors and it is especially effective on the last class, whilst still retaining high accuracy on the past ones thanks to the regularization brought in by background inpainting and replay strategies. S&R, instead, heavily forgets previous classes, thus confirming the usefulness of replay data.

Addition of last 5 classes. In this setup, 15 classes are learned in the initial step, while the remaining 5 are added in one shot (15-5) or sequentially one at a time (15-1). Compared to the 19-1 setup, the addition of multiple classes in the incremental iterations makes catastrophic forgetting even more severe. The accuracy gap between FT and joint training, in fact, raises from about 41% of the 19-1 case to more than 70% of mIoU in the 15-1 scenario. Taking a closer look at the results in Table 5.1 (upper mid and right parts), our replay approaches strongly limit the degradation caused by catastrophic forgetting. This trend can be observed in the 15-5 setup and more evidently in the 15-1 one, both in the disjoint and overlapped settings: exploiting generated or web-derived replay samples proves to effectively restore knowledge of past classes, leading to a final mIoU approaching that of the joint training. Storing and replaying original samples, instead, improves the performance with respect to FT, but ultimately leads to a mIoU lower of more than 20% if compared to our approaches. This is due to the limited number of samples to be stored in order to match the helper decoder size: their sole addition is, in fact, insufficient to adequately preserve learned knowledge. Finally, we observe that RECALL can scale much better than competitors when multiple incremental steps are performed (scenario 15-1), as typically encountered in real-world applications.

Addition of last 10 classes. To analyze the previous claim, we introduce some new challenging experiments, not evaluated in previous works. In these tests only 10 classes are observed in the initial step, while the remaining ones are added in a single batch (10-10), in 2 steps of 5 classes each (10-5), or individually (10-1). Again, FT is heavily affected by the information loss that oc-

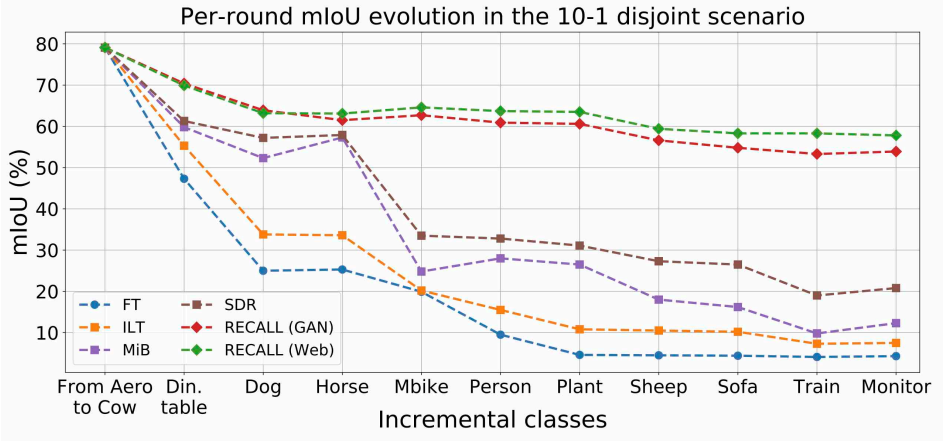


Figure 5.4: Evolution of mIoU on the 10 tasks of 10-1 disjoint.

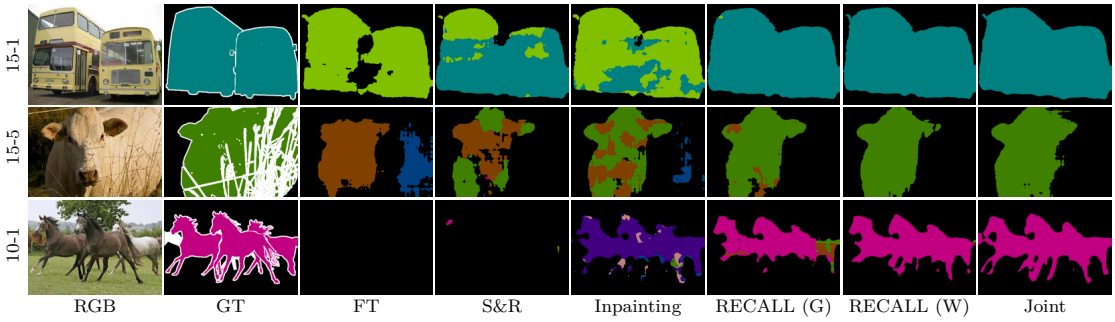


Figure 5.5: Qualitative results on disjoint incremental setups: from top to bottom 15-1, 15-5 and 10-1.

curs when performing incremental training without regularization, leading to performance drops up to about 71% of mIoU with respect to the joint training in the most challenging 10-1 setting. Thanks to the introduction of replay data, RECALL brings a remarkable performance boost to the segmentation accuracy and becomes more and more valuable as the difficulty of the settings increases. In the 10-10 case, our method achieves slightly lower mIoU results than competitors (although comparable). As we increase complexity, our approach is able to outperform competitors by about 8% of mIoU in 10-5 and by 37% of mIoU in 10-1. We remark how RECALL shows a convincing capability of providing a rather steady accuracy in different setups, regardless of the number of incremental steps used to introduce new classes. For example, in the disjoint scenario, when moving from simpler to more challenging setups (*i.e.*, from 10-10 to 10-1, passing through 10-5), the mIoU of FT drops as $33.0\% \rightarrow 23.7\% \rightarrow 4.3\%$ and the one of SDR (*i.e.*, the best compared approach) as $62.9\% \rightarrow 52.0\% \rightarrow 20.8\%$, while our approach maintains a stable mIoU trend of $61.9\% \rightarrow 60.6\% \rightarrow 57.8\%$. Finally, we report the mIoU after each incremental step on the 10-1 disjoint scenario in Figure 5.4, where our approaches show much higher mIoU at every learning step, indicating improved resilience to forgetting and background shift, than competitors.

In the qualitative results in Figure 5.5 we observe that RECALL effectively alleviates forgetting and reduces the bias towards novel classes. In the first row, the *bus* is correctly preserved while FT, S&R and inpainting wrongly classify it as *train* (*i.e.*, one of the novel classes); in the

second row, FT places *sheep* and *tv* (newly added classes) in place of *cow*; in the third row, some *horse*'s features are either mixed with those of *person* and *cat* or completely destroyed, while they are preserved by our methods (the web scheme shows higher accuracy than the GAN here).

5.6.2 Qualitative Results

We report some qualitative results: Figure 5.6 displays the segmentation output in the disjoint scenario for all the experimental incremental training protocols (*i.e.*, FT, background inpainting, RECALL with GAN or Web and joint). In particular, we show the results for a couple of samples in each of the 6 considered setups (*i.e.*, 19-1, 15-5, 15-1, 10-10, 10-5 and 10-1). Finally, Figure 5.7 shows the evolution of the output maps across the incremental steps in the 15-1 scenario for a couple of sample images.

In Figure 5.6 it is possible to see that the background inpainting strategy constitutes a clear improvement with respect to the simple fine-tuning approach, allowing to reduce catastrophic forgetting, which is very critical in FT. However, forgetting is still fairly noticeable with the sole inpainting strategy, where the output maps are quite noisy and relevant parts of the objects get lost in many scenes, typically overestimating the background class. The addition of replay data in both the GAN and the Web-based solutions proves to be very effective in further reducing the forgetting phenomenon, thus providing a final segmentation performance very close to the joint-training reference except for some details, which are typically close to the boundaries of the objects. Furthermore, our approaches do not mislead previous classes with similar ones introduced in the incremental steps (*e.g.*, FT and inpainting mislead the *cow* with *sheep* in row 3 and the *bus* with *train* in row 4).

The accuracy boost introduced by the proposed replay strategies can be further appreciated in Figure 5.7, where we report the segmentation output computed after each incremental step of the 15-1 disjoint setup for a couple of image samples. The improvement can be noted by looking, for example, at the images on the second and fourth incremental steps, where the new classes *sheep* and *train* are introduced respectively. When FT or background inpainting are adopted, the segmentation network tends to experience a severe forgetting of the old classes *cow* and *bus* (which are mistaken for visually similar novel ones). This behavior is corrected by providing replay training data to the network: both GAN and Web-based strategies are able to preserve an accurate recognition of old classes, even when semantically similar ones are incrementally added.

5.6.3 Ablation Studies

To further validate the robustness of our approach, we perform some ablation studies. First of all, we analyze the memory requirements. The plot in Figure 5.8 shows in semi-log scale the memory occupation (expressed in MB) of the data to be stored at the end of each incremental step, as a function of the number of classes learned up to that point. We denote with *standard* an incremental approach which do not store any sample (*e.g.*, FT, LwF, ILT, MiB, SDR). The saved model generally corresponds to a fixed size encoder and a decoder, whose dimension slightly increases at each step to account for additional output channels for the new learnable classes. *Saving images*, instead, refers to the extreme scenario where training images of past steps are stored, thus being available throughout the entire incremental process. As concerns our approach, to annotate originally weakly-labeled replay images, we devise a specific module (Section 5.3), which requires to save a set of helper decoders $\{D^H_{\mathcal{C}_i}\}_{i=0}^{k}$, one for each past step. Finally, for the GAN-based approach we add the storage required for the generative model. Figure 5.8 shows that our web-based solution is very close to the *standard* ones in terms of

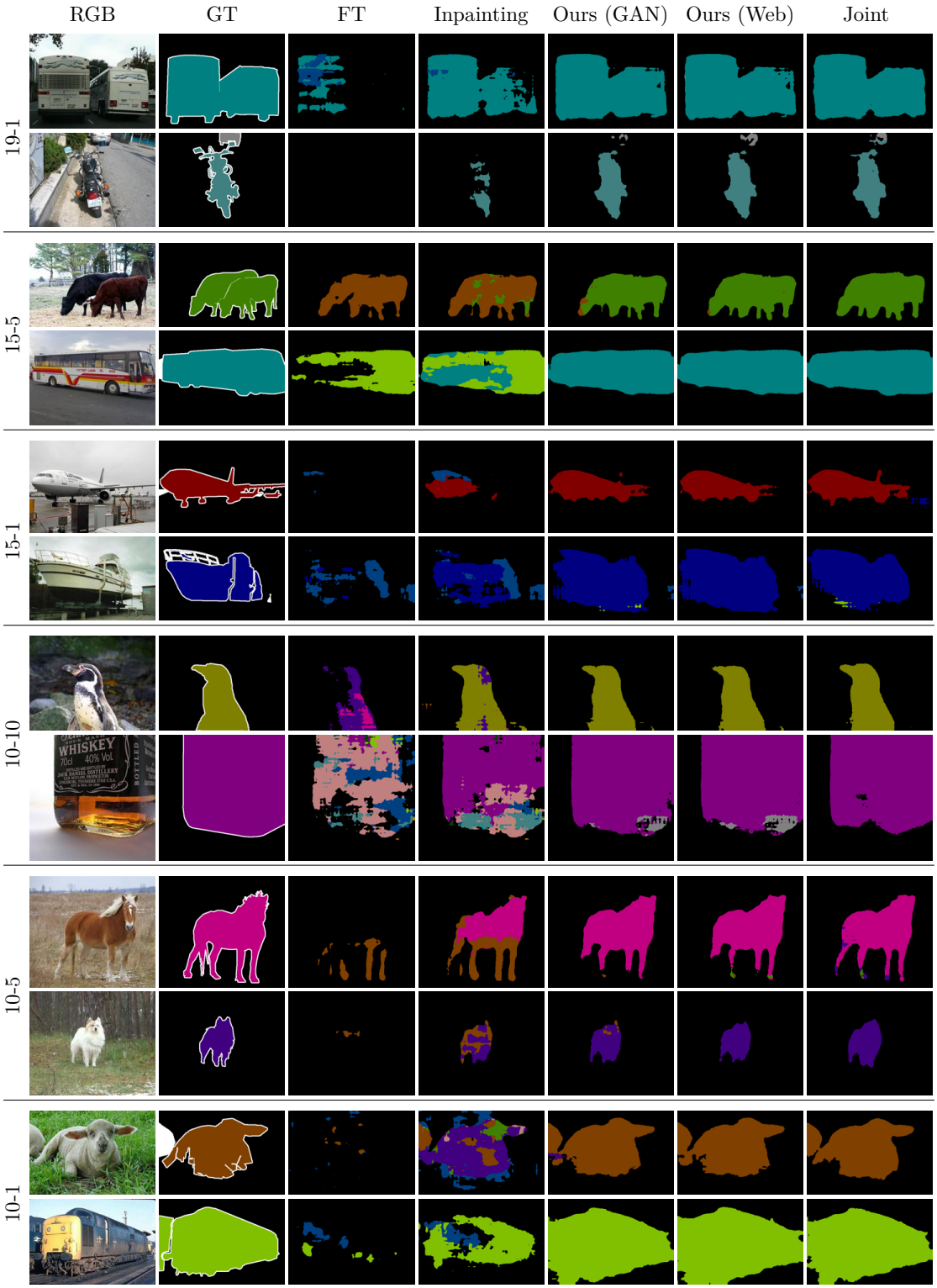


Figure 5.6: Qualitative results on disjoint incremental setups.

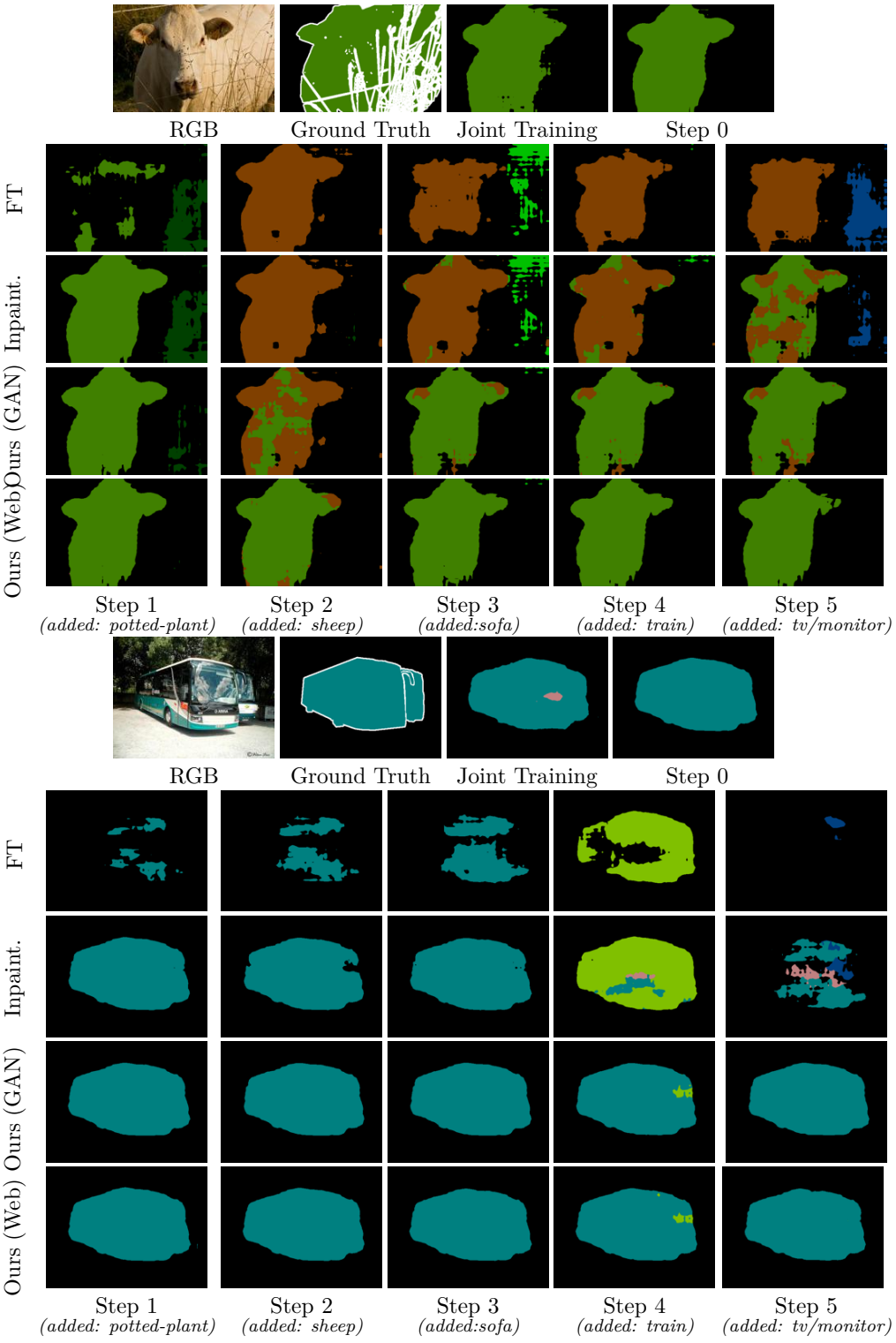


Figure 5.7: Per-step prediction maps on the 15-1 disjoint incremental setup for different training strategies.

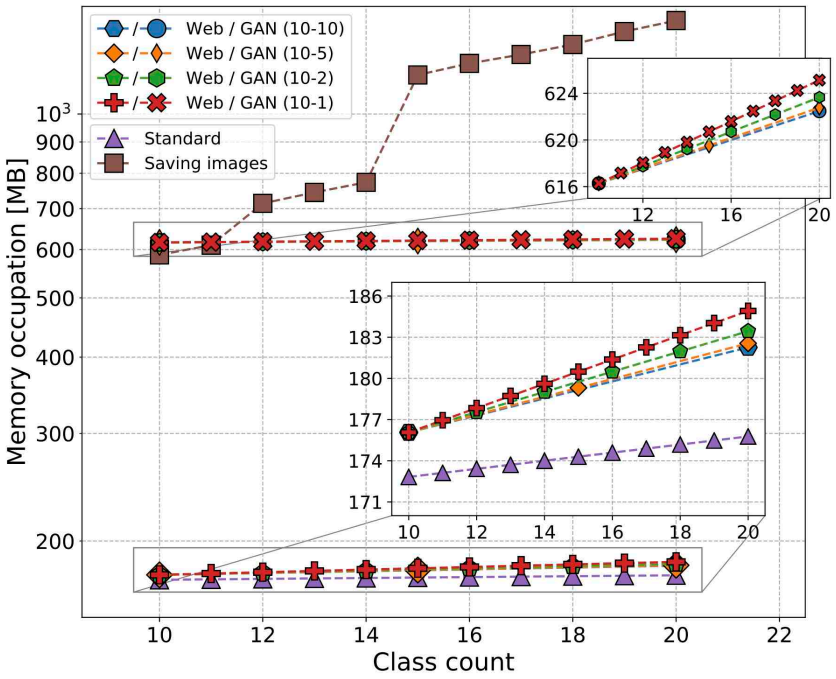


Figure 5.8: Memory occupation in the disjoint scenario.

Table 5.2: mIoU results showing the contribution of each module, D: Disjoint, O: Overlapped.

	19-1		15-5		15-1		10-10		10-5		10-1	
Method	D	O	D	O	D	O	D	O	D	O	D	O
Bgr inp.	65.6	66.7	52.2	52.5	49.7	49.9	58.8	60.7	47.5	47.1	34.0	39.0
GAN	54.5	56.2	49.8	49.1	47.9	48.2	45.8	48.8	38.1	43.7	36.6	40.8
Web	57.3	57.4	55.2	54.7	55.0	53.7	55.2	58.2	47.9	52.1	45.4	50.1
GAN+inp.	65.8	68.4	63.5	64.0	62.1	62.7	60.8	63.1	57.8	58.4	53.9	54.8
Web+inp.	65.4	68.6	66.3	65.6	64.3	64.8	61.9	63.7	60.6	62.3	57.8	60.7

memory occupation. The space required to store the GAN is comparable to that needed to save images in the very initial steps, but then remains constant while the space for saving all training data quickly grows.

We further analyze the contribution of the background inpainting and replay techniques in Table 5.2. While inpainting alone provides a solid contribution in terms of knowledge preservation acting similarly to knowledge distillation, we observe that its effect tends to attenuate with multiple incremental steps. For example, moving from 10-10 to 10-1 overlapped setups, the mIoU drops more than 20%. On the other hand, the proposed replay techniques prove to be beneficial when multiple training stages are involved. On the same setting, replay techniques alone limit the degradation to only 8%. Yet, jointly employing replay and inpainting further boosts the final results in all setups (up to 15%), proving that they can be effectively combined.

Finally, we analyze how results vary with respect to the proportion of new (r_{new}) and replay (r_{old}) samples seen during training (Figure 5.9): the mIoU is quite stable with respect to this ratio, however the maximum value is reached when the same number of old and replay samples

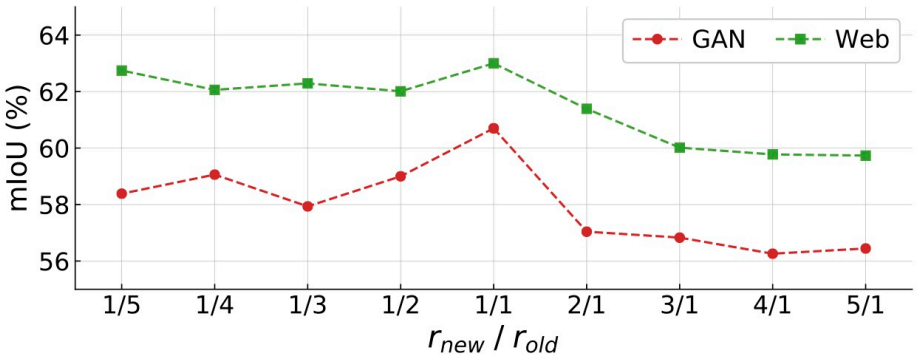


Figure 5.9: Comparison of different interleaving policies in 15-1 disjoint.

Table 5.3: Mean IoU achieved by the proposed approach on the Pascal VOC2012 dataset for different incremental setups and pre-training strategies.

		19-1						**15-5**						**15-1**					
		Disjoint			**Overlapped**			**Disjoint**			**Overlapped**			**Disjoint**			**Overlapped**		
Method	**Init**	1-19	20	all	1-19	20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all
GAN	ImageNet	65.2	50.1	65.8	67.9	53.5	68.4	66.3	49.8	63.5	66.6	50.9	64.0	66.0	44.9	62.1	65.7	47.8	62.7
	MSCOCO	68.7	58.4	69.3	68.6	59.6	69.3	70.3	58.4	68.5	70.3	59.5	68.7	70.4	55.5	67.8	70.8	57.5	68.6
Web	ImageNet	65.0	47.1	65.4	68.1	55.3	68.6	69.2	52.9	66.3	67.7	54.3	65.6	67.6	49.2	64.3	67.8	50.9	64.9
	MSCOCO	69.7	55.3	70.1	68.8	60.7	69.5	70.7	59.2	69.0	70.7	59.9	69.1	70.9	57.4	68.7	71.2	55.7	68.5

		10-10						**10-5**						**10-1**					
		Disjoint			**Overlapped**			**Disjoint**			**Overlapped**			**Disjoint**			**Overlapped**		
Method	**Init**	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all
GAN	ImageNet	62.6	56.1	60.8	65.0	58.4	63.1	60.0	52.5	57.8	60.8	52.9	58.4	58.3	46.0	53.9	59.5	46.7	54.8
	MSCOCO	68.2	64.5	67.6	68.3	66.1	68.4	68.2	62.1	66.4	67.3	61.9	65.8	68.2	58.3	64.6	67.8	60.5	65.4
Web	ImageNet	64.1	56.9	61.9	66.0	58.8	63.7	63.2	55.1	60.6	64.8	57.0	62.3	62.3	50.0	57.8	65.0	53.7	60.7
	MSCOCO	68.0	64.9	67.7	68.2	66.4	68.4	68.6	63.9	67.4	67.6	64.6	67.3	68.0	58.0	64.3	68.5	62.5	66.7

is used, *i.e.*, $r_{new}/r_{old} = 1$.

5.6.4 Analyses on Pre-Training

The weights of semantic segmentation deep learning architectures are typically initialized with pre-trained values computed on a large dataset for a related (but usually different) task. The most common pre-training strategy consists in using weights computed on image classification large-scale datasets, such as ImageNet [121]. On the other hand, it has been shown [5, 20] that pre-training weights on a related segmentation dataset, such as MS COCO [120], could further boost results on semantic segmentation benchmarks. However, using another semantic segmentation dataset raises some concerns about the fact that the pre-training data could contain information about the tasks to be learned in the incremental steps, thus following previous works [18, 21, 23] we decided for a more conservative approach using ImageNet pre-training.

To further investigate this aspect we show extensive results in Table 5.3 comparing ImageNet and MS COCO pre-training strategies for all the considered incremental setups. Here we can see that, as expected, pre-training on MS COCO always helps incremental semantic segmentation. We argue that the motivation is at least two-fold: first, a better initialized model on the same target task could converge to a better solution, and second, a model pre-trained on MS COCO

could have already learned some spatial and semantic information of classes added to the model in incremental steps (this second point is the reason why we decided to avoid using this strategy even if it leads to better results). More in detail, we can observe that incremental approaches show significant improvements of up to 15%. This quite large gap could be due also to the encoder freezing procedure we employ in our our work (similarly to [18, 20]) that reduces catastrophic forgetting, but at the same time does not allow the network to update the feature extraction module according to the information in the samples of the new classes at each incremental step, which can be used only to update the decoder. Indeed, a model initialized with pre-trained weights on MS COCO (which has enough variability and semantic content information) needs less training steps to adapt to a new (related) semantic segmentation dataset (*i.e.*, different domain but same task). On the other side, pre-training on a different task and different domain (*e.g.*, on ImageNet) requires more training steps to adapt to the new scenario. We can verify this claim looking at Table 5.3, where we can observe how the mIoU gap between the two pre-training strategies is larger when the initial step has fewer classes (*e.g.*, 10).

5.6.5 Class Mapping Module

Here we provide some further analysis and insights on the Class Mapping Module (introduced in Section 5.3), which is used to translate each class of the semantic segmentation incremental dataset (*e.g.*, Pascal VOC2012 [114]) to the most similar class of the GAN's training dataset (*e.g.*, ImageNet [121]). Notice that properly mapping the labels between the different domains is an important step, since incorrect pairings may easily harm the accuracy of the final model.

To solve the task, we took an Image Classifier I pre-trained to address an image classification task on the GAN's dataset. Then, for each class c in the current label set we select the corresponding training subset (*i.e.*, all the samples of the current training set associated to class c), and we sum the resulting class probability vectors from the classification output (according to I). An $\arg\max$ operation is then performed, to identify the GAN's class c_G with the highest probability score. To show the effectiveness of the proposed classification, we report in Table 5.4 the 3 classes from the GAN's dataset with the highest score for each class of the Pascal VOC2012 dataset. We can see that for all the classes the top selected pairings appear reasonable at first (notice that only the best matching class is selected in the proposed approach). At a closer look, we find that the classifier selects an unexpected label only in a single case, that is the *person* class being translated into *cowboy hat*; however, we remark that the ImageNet dataset does not contain the *person* class, thus inherently lacking a close match for that category. In light of this, we believe that the chosen class (*i.e.*, *cowboy hat*) is a reasonable choice and may still help in retaining high accuracy on the *person* class, being the *cowboy hat* always shown on top of people's heads. This situation is interesting as it shows the robustness of our approach not only to different domains with different statistical distributions (ImageNet domain versus Pascal VOC2012 one), but also to different labeling domains (the label set of VOC2012 is not a subset of the ImageNet one).

The sample generated images in Figure 5.10 allow to verify the effectiveness of the conditioned image generation strategy. Notice how in most cases the images are very similar to the Pascal VOC ones, even for the *person* class that does not have a direct mapping. For the sake of comparison, the figure also reports two randomly sampled images for each class taken either from the Pascal VOC2012 dataset (first two columns of Figure 5.10) or from Flickr for the Web approach (last two columns of Figure 5.10).

Figure 5.10: Original images from the incremental Pascal VOC dataset, together with replay data generated by GAN or retrieved by Flickr's web crawler. From top to bottom: *airplane*, *train*, *bicycle*, *person*, *bird*, *cow*, *horse*, and *sheep*.

Table 5.4: Class mapping between Pascal VOC and ImageNet datasets. The table shows the 3 best matching ImageNet classes for each Pascal VOC2012 class. (*): matching classes for *tv/monitor* are not computed since replay data is not needed.

Pascal		**ImageNet**		
index	class	1st class	2nd class	3rd class
1	airplane	airliner	warplane	wing
2	bicycle	mountain bike	tandem	tricycle
3	bird	kite (bird)	dipper	quail
4	boat	catamaran	lakeside	fireboat
5	bottle	beer bottle	soda bottle	water bottle
6	bus	trolleybus	carriage	minibus
7	car	racing car	station wagon	minivan
8	cat	tabby cat	Egyptian cat	tiger cat
9	chair	rocking chair	dining table	folding chair
10	cow	ox	oxcart	water ox
11	dining table	dining table	china closet	restaurant
12	dog	Labrador retriever	pit bull terrier	beagle
13	horse	sorrel	ox	fox squirrel
14	motorbike	moped	scooter	disc brake
15	person	cowboy hat	crash helmet	crutch
16	potted plant	pot	pencil case	greenhouse
17	sheep	ram	llama	bighorn sheep
18	sofa	studio couch	quilt	rocking chair
19	train	carriage	electric locomotive	freight car
20	tv/monitor*	-	-	-

5.6.6 Per-Class Quantitative Results

For a more detailed evaluation, we present the per-class IoU values for some of the proposed approaches and scenarios. We considered the following methods in the disjoint scenario on all the experimental protocols: fine-tuning (FT), background inpainting, RECALL (GAN), RECALL (Web) and joint training. The results are summarized in Table 5.5. From here, we can appreciate how fine-tuning always catastrophically forgets previous classes when learning new ones. The simple background inpainting strategy allows to largely alleviate such phenomenon bringing a similar effect to recent knowledge distillation approaches [20, 23]. On top of this, we apply GAN or Web-based replay strategies to regularize training and background content inpainting scheme to reduce bias toward the background. While these strategies are specifically designed to preserve old knowledge, they also allow to achieve large mIoU gains on new classes reducing the false positive rate (*i.e.*, the detection of new classes in locations containing the old ones).

In order to better understand the effect of our proposed modules, we report in Table 5.6 the Pixel Accuracy (PA) and the IoU for the class being added at each step of the disjoint 10-1 scenario. The results demonstrate that, on the newly introduced class, FT generally achieves a very high PA (top-left) and a per-class IoU (top-right) comparable to the other approaches. Yet, FT concurrently shows very low mIoU over all classes learned up to the current step (bottom-right), as well as over only previously seen categories (bottom-left). All combined, this is indicative of

Table 5.5: Per-class IoU of compared methods in disjoint experimental protocol on multiple scenarios of Pascal VOC2012.

	Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
19-1	FT	72.4	62.4	6.7	45.0	47.1	39.5	33.7	40.9	25.7	4.3	54.0	8.0	25.0	50.4	50.6	0.0	35.3	43.0	0.8	59.5	13.2	35.2	13.2	34.2
	Inpainting	91.0	83.9	35.1	77.3	62.3	70.7	77.9	73.4	85.7	31.5	73.1	48.0	81.3	74.4	64.6	81.0	44.1	75.7	41.3	74.5	30.4	66.1	30.4	65.6
	GAN	91.7	82.8	32.3	82.6	62.8	74.1	86.2	79.6	86.0	30.0	58.9	45.9	80.5	67.9	73.4	80.6	35.3	62.9	39.6	77.9	50.1	65.2	50.1	65.8
	Web	91.4	82.8	35.9	83.4	59.9	73.5	85.3	73.7	85.7	31.3	59.4	40.9	81.1	67.1	73.4	80.5	43.1	61.5	42.6	74.4	47.1	65.0	47.1	65.4
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	75.5	73.5	75.4
15-5	FT	72.4	62.4	6.7	45.0	47.1	39.5	33.7	40.9	25.7	4.3	54.0	8.0	25.0	50.4	50.6	0.0	35.3	43.0	0.8	59.5	13.2	35.2	13.2	34.2
	Inpainting	89.0	68.7	36.0	68.2	48.4	71.4	12.8	77.3	85.6	26.7	8.1	48.8	80.3	61.6	68.8	78.7	20.1	29.0	26.3	38.4	51.8	56.1	33.1	52.2
	GAN	90.4	78.8	35.0	79.5	60.3	75.7	79.3	78.7	85.9	22.8	55.0	46.6	80.0	67.4	72.1	77.8	37.3	60.2	32.2	64.4	55.1	66.3	49.8	63.5
	Web	90.8	82.2	35.5	81.7	63.9	75.3	85.0	77.8	86.3	28.0	67.5	48.7	81.0	72.7	73.8	78.0	40.4	65.7	31.9	69.1	57.6	69.2	52.9	66.3
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4
15-1	FT	74.2	27.2	0.0	1.6	15.1	11.3	0.0	4.1	0.5	0.0	0.0	0.0	0.0	0.2	0.2	0.0	27.0	25.6	28.9	33.5	52.2	8.4	33.5	14.4
	Inpainting	85.9	38.9	31.4	79.4	41.5	71.3	28.9	62.6	85.6	32.2	29.6	50.2	76.6	69.2	55.3	80.2	18.5	37.4	36.3	19.8	17.9	55.5	26.0	49.9
	GAN	90.5	80.7	34.5	79.5	59.1	75.5	72.7	78.2	85.3	25.3	59.0	39.9	79.9	68.8	72.5	78.6	23.2	58.0	39.2	60.1	43.8	66.0	44.9	62.1
	Web	90.5	82.1	34.4	81.5	62.6	76.0	82.3	77.0	85.1	27.4	63.6	39.4	80.3	71.9	72.2	78.4	35.4	64.4	35.7	61.9	48.7	67.6	49.2	64.3
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4
10-10	FT	82.1	0.2	0.0	1.2	0.0	1.4	0.0	0.0	0.0	0.0	0.0	52.3	73.2	49.8	73.1	81.8	41.4	49.7	49.1	76.1	62.2	7.7	60.9	33.0
	Inpainting	90.9	81.8	34.1	73.1	58.6	73.3	85.6	78.8	78.2	29.0	29.1	43.7	66.6	47.7	73.0	74.2	29.6	57.3	38.8	70.9	61.4	62.2	56.3	60.7
	GAN	90.8	83.3	30.4	75.8	61.4	73.5	80.8	77.2	72.8	23.6	46.8	48.0	65.4	55.3	66.1	72.5	36.8	58.3	36.1	67.1	55.6	62.6	56.1	60.8
	Web	90.9	82.3	32.7	75.4	63.2	72.8	81.7	73.5	76.2	24.2	58.5	46.5	68.8	60.2	64.7	73.3	38.3	58.3	34.2	68.5	56.2	64.1	56.9	61.9
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4
10-5	FT	78.2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	16.4	13.7	23.4	55.7	46.7	37.4	39.8	47.9	75.6	62.3	7.2	41.9	23.7
	Inpainting	88.5	58.8	31.9	55.4	58.2	69.2	0.2	78.3	83.2	28.2	5.0	36.4	71.6	34.7	61.4	74.1	20.4	26.2	25.5	34.0	47.9	46.8	43.2	47.1
	GAN	89.3	77.9	28.9	72.1	59.3	73.6	75.1	75.8	79.5	20.5	37.2	44.2	67.8	50.9	59.5	71.3	31.4	51.9	32.3	63.1	53.0	60.0	52.5	57.8
	Web	89.5	80.8	31.2	74.6	61.6	72.0	81.6	74.3	80.6	19.8	55.1	44.3	69.2	56.9	56.5	71.7	39.8	59.0	30.2	69.7	54.2	63.2	55.1	60.6
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4
10-1	FT	69.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19.9	6.3	2.0	4.3
	Inpainting	85.0	35.3	28.6	72.6	37.2	67.9	16.9	65.3	83.0	32.1	13.3	35.1	41.4	5.0	22.5	71.7	21.5	16.9	34.5	19.2	13.0	45.2	28.1	39.0
	GAN	88.8	77.9	26.6	71.8	58.6	73.2	63.5	74.0	75.7	20.5	41.3	34.2	60.5	43.6	58.6	66.2	15.8	51.7	37.4	53.0	38.8	58.3	46.0	53.9
	Web	89.1	79.1	31.0	74.4	62.2	66.5	81.7	74.1	78.7	19.4	56.2	41.8	62.7	58.5	62.1	66.6	8.5	59.3	36.8	59.2	45.0	62.3	50.0	57.8
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4

Table 5.6: Per-round accuracy measures in the 10-1 disjoint scenario. In the top part we report the PA (left) and IoU (right) of the last class currently introduced. The bottom part, instead, shows the mean IoU over the old classes up to the ongoing step (left), as well as the overall mean IoU including the new classes (right). The classes added at each incremental step are: 1:*dining table*, 2:*dog*, 3:*horse*, 4:*motorbike*, 5:*person*, 6:*potted plant*, 7:*sheep*, 8:*sofa*, 9:*train* and 10:*tv/monitor*. Best in **bold**.

PA (new)	*step 1*	*step 2*	*step 3*	*step 4*	*step 5*	*step 6*	*step 7*	*step 8*	*step 9*	*step 10*
FT	69.2	**90.9**	**87.1**	**79.2**	88.3	3.8	26.9	**49.9**	55.0	57.5
ILT [18]	70.2	88.3	44.3	52.8	86.9	58.7	22.5	43.6	48.8	56.3
MiB [23]	**75.5**	87.2	38.5	51.2	**89.6**	**61.0**	6.1	38.2	47.4	**60.2**
SDR [21]	68.7	73.2	34.0	31.0	84.5	55.6	15.7	39.0	45.1	55.8
ours (GAN)	24.0	56.7	38.1	58.5	76.2	27.1	36.8	25.4	53.0	45.0
ours (Web)	23.8	57.3	46.1	62.3	79.3	36.9	**42.8**	26.9	**64.2**	57.4

IoU (new)	*step 1*	*step 2*	*step 3*	*step 4*	*step 5*	*step 6*	*step 7*	*step 8*	*step 9*	*step 10*
FT	16.2	37.0	15.8	12.6	**78.4**	3.2	9.2	16.3	9.7	19.9
ILT [18]	14.6	37.2	14.5	11.5	68.6	5.5	4.9	10.3	11.1	13.8
MiB [23]	14.6	38.0	12.2	9.5	66.0	9.6	1.7	5.5	10.9	7.0
SDR [21]	**25.0**	54.0	13.3	10.9	69.2	10.9	5.4	8.9	15.4	17.8
ours (GAN)	22.4	53.9	36.1	54.6	64.4	23.6	34.4	23.6	49.3	38.8
ours (Web)	22.1	**54.5**	**43.7**	**58.4**	67.9	**30.5**	**39.6**	**24.7**	**58.3**	**45.0**

mIoU (old)	*step 1*	*step 2*	*step 3*	*step 4*	*step 5*	*step 6*	*step 7*	*step 8*	*step 9*	*step 10*
FT	50.4	23.9	26.1	20.5	4.6	4.7	4.2	3.7	3.8	3.5
ILT [18]	59.4	33.5	35.2	20.9	11.7	11.2	10.9	10.2	7.1	7.2
MiB [23]	64.3	53.6	61.1	26.0	25.3	27.6	19.0	16.8	9.7	12.6
SDR [21]	64.9	57.5	61.6	35.2	30.2	32.4	28.7	27.5	19.2	21.0
ours (GAN)	**74.7**	**64.7**	63.4	63.2	60.7	62.9	57.9	56.5	53.5	54.6
ours (Web)	74.3	63.9	**64.6**	**65.0**	**63.5**	**65.5**	**60.6**	**60.1**	**58.3**	**58.4**

mIoU (all)	*step 1*	*step 2*	*step 3*	*step 4*	*step 5*	*step 6*	*step 7*	*step 8*	*step 9*	*step 10*
FT	47.3	25.0	25.3	19.9	9.5	4.6	4.5	4.4	4.1	4.3
ILT [18]	55.3	33.8	33.6	20.2	15.5	10.8	10.5	10.2	7.3	7.5
MiB [23]	59.8	52.3	57.3	24.8	28.0	26.5	18.0	16.2	9.8	12.3
SDR [21]	61.3	57.2	57.9	33.5	32.8	31.1	27.3	26.5	19.0	20.8
ours (GAN)	**70.4**	**63.9**	61.5	62.7	60.9	60.6	56.6	54.8	53.3	53.9
ours (Web)	69.9	63.2	**63.1**	**64.6**	**63.7**	**63.5**	**59.4**	**58.3**	**58.3**	**57.8**

Table 5.7: mIoU on VOC2012 disjoint 15-1 with replay data. G: GAN, F: Flickr. Naïve: only decoder of last step is used for pseudo-labeling, ours: our complete approach (RECALL) is used.

	none	+ naïve (G)	+ naïve (F)	+ ours (G)	+ ours (F)
ILT	5.4	37.8 (+32.4)	39.9 (+34.5)	49.6 (+44.2)	51.5 (+46.1)
MiB	37.9	49.3 (+11.4)	50.5 (+12.6)	63.5 (+25.6)	65.7 (+27.8)
SDR	48.1	53.1 (+05.0)	55.8 (+07.7)	65.5 (+17.4)	66.5 (+18.4)

an overestimation of the new class. In other words, FT progressively forgets foregoing semantic information, while predicting more often the newly seen class (which experiences high PA but low IoU, due to many false positive predictions). Our approach, instead, can effectively improve knowledge preservation thanks to replay data and background inpainting, providing steady mIoU results throughout the incremental steps.

5.6.7 Combining RECALL with Other Techniques

To the best of our knowledge, no works on continual semantic segmentation using GAN-generated or web-crawled data exist. The aim of our work is to provide a general framework to retrieve and employ unlabeled replay data. In this section, we demonstrate that our framework can be applied on top of competing approaches to improve their performance: some experimental results are shown in Table 5.7. Adding replay data with naïve pseudo-labeling (*i.e.*, using the decoder of the previous step) already leads to a performance improvement, but combining our method with previous approaches leads to much higher results with improvements ranging from 17% to 46%, proving the effectiveness and general applicability of the modules introduced in RECALL.

5.6.8 Preliminary Analyses on ADE20K

Up to this point, we reported only experiments on Pascal VOC2012, which contains object-level classes. However, when addressing real-world tasks, the complete understanding of the

surroundings is usually required: for instance, to distinguish a mixture of *stuff* and object-level classes, as in the ADE20K dataset [117]. Indeed, the ADE20k dataset poses a great challenge due to the vast class set, comprising *stuff* categories not present in Pascal VOC2012. We remark that exact correspondence between GAN's conditioning class space and semantic segmentation category set is not required by our replay strategy. For example, as we have already observed, the *person* class is missing in the ImageNet dataset (used for pre-training the generative model), but images of people can still be retrieved from generated images of semantically related categories, such as *hat* (see Table 5.4). Thus, even when the generative model cannot be explicitly conditioned to reproduce some specific segmentation classes (*e.g.*, *stuff* categories), it is possible to retrieve instances of them just relying on semantically correlated categories. This retrieval (*i.e.*, mapping) operation is performed automatically by our approach. This is even more true for the web-replay strategy, where we have complete control over the keywords used for the search.

Hence, we argue that our approach is suitable for continual semantic segmentation even when non-object categories are present. To make our point, we run preliminary experiments on ADE20k. We consider the 100-10 setting, where 100 classes are learned in the first step and the others are added in batches of 10 at each incremental step. The FT baseline reaches a very low mIoU of 0.8%, while our RECALL with GAN-based replay samples improves the score up to 11.4%, showing that our methods mitigate catastrophic forgetting even in this challenging setup. To achieve these results, we did not perform any parameter tuning (*i.e.*, we kept the same pre-processing, learning parameters, batch and image sizes used for VOC2012). However, further experiments and tuning are needed to provide a proper comparison with other works on this benchmark.

5.7 Conclusions

In this chapter we introduced RECALL, which targets continual semantic segmentation by means of replay strategies to alleviate catastrophic forgetting and background inpainting to mitigate background shift. Two replay schemes are proposed to retrieve data related to former training stages, either reproducing it via a conditional GAN or crawling it from the web. The experimental analyses proved the efficacy of our framework in improving accuracy and robustness to multiple incremental steps compared to competitors. Future research will improve the generative model, coupling it more strictly with the incremental setup, and explore how to control and refine weak supervision during web-crawling. Extensive evaluation on different datasets, such as ADE20K, will also be performed.

5.8 Final Remarks

This part of the dissertation on continual semantic segmentation started from its definition and problem setup in Chapter 2, then we successfully explored knowledge distillation to preserve the output on previously seen classes in Chapter 3 and latent space regularization approaches to prevent overfitting on the novel classes in Chapter 4. In this chapter, instead, we showed how we can embed (additional) replay data into any continual learning framework.

Future work will be devoted towards a more precise integration of all the techniques we have explored in these chapters. Indeed, being able to effectively combine knowledge distillation at the output and feature level with latent space regularization and replay-based techniques is still an open research direction with great potential, where the combined positive effects of each of the strategies could outperform current state-of-the-art approaches.

Part II

Coarse-to-Fine Learning of Semantic Categories

6

Coarse-to-Fine Learning of Semantic Concepts

In this chapter we slightly change the focus with respect to the previous ones. In the previous chapters we have defined and unveiled techniques to solve class-incremental (*i.e.*, continual) learning in semantic segmentation, where a deep learning model trained on an initial data distribution with a starting set of classes is updated to incrementally recognize new semantic categories given new samples associated to them. In this chapter, instead, we define and tackle the coarse-to-fine learning of new semantic concepts. A deep learning model is trained on an initial set of classes with a given training set of samples; in a latter stage, the model is updated to recognize either more refined categories than those originally introduced, *e.g.*, from a generic *object* class to *table* (we call it *semantic level* coarse-to-fine learning [24], see Section 6.2), or sub-parts belonging to the object-level class (we call it *spatial level* coarse-to-fine learning [25], see Section 6.3)

6.1 Introduction

We have already observed in Chapter 1 how different computer vision applications demand for different levels of semantic understanding of the surroundings (*i.e.*, from image classification, to object detection and location to semantic segmentation). Even within the semantic segmentation task, different levels of granularity can be defined (*e.g.*, *dog leg*, *dog* or *animal*). However, it is common practice to train a generic base model on a generic set of classes which can solve some initial tasks. Again, sometimes the focus of the application changes in a later stage, after deployment has already been accomplished.

In the previous chapters we have observed how we can handle and adapt to changes in the set of semantic categories recognized by the deep learning architecture (*i.e.*, continual learning of new semantic concepts over time). In this chapter, we take a different perspective and we aim at updating a pre-trained model on a generic set of classes in order to recognize more specific semantic categories and at making use of the coarse-level model to leverage the accuracy on the finer split of classes. The task presented in this chapter is considerably different from that of previous chapters in that all the training images are available from the beginning and the sets of labels are progressively split into more fine-grained hierarchical categories and then fed to the learner, as can be appreciated in Figure 6.1.

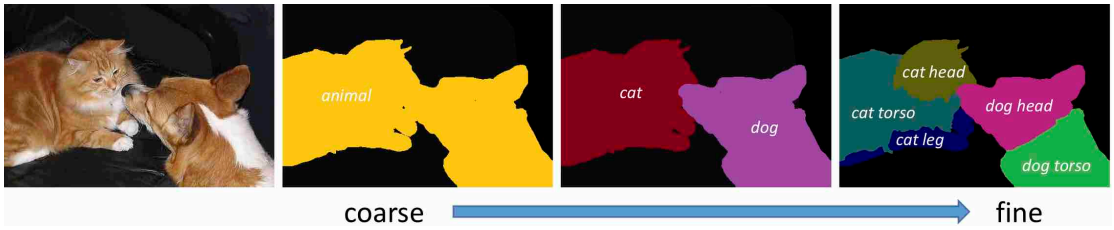


Figure 6.1: Different levels of semantic image understanding: from coarse to fine.

6.1.1 Contributions

Our main intuition is that we could exploit the learned model on the coarse set of classes to aid the learning process of the more specific model, achieving higher results. This is, in some sense, similar to curriculum learning [164] applied to coarse-to-fine semantic sets.

In this chapter, we distinguish between two different coarse-to-fine learning setups: namely, at the semantic level [24] (Section 6.2) and at the spatial level [25] (Section 6.3).

Nonetheless, the two scenarios are conceptually similar:

- **semantic coarse-to-fine-learning**: in this case, macro-level classes (*e.g.*, *furniture*) are split to generic object-level classes (*e.g.*, *table*) and finally to specific classes (*e.g.*, *dining table*). This problem is defined in Section 6.2: we address the multi-level semantic segmentation task where a deep neural network is first trained to recognize an initial, coarse, set of a few classes. Then, in an incremental-like approach, it is adapted to segment and label new objects' categories hierarchically derived from subdividing the classes of the initial set. We propose a set of strategies where the output of coarse classifiers is fed to the architectures performing the finer classification. Furthermore, we investigate the possibility to predict the different levels of semantic understanding together, which also helps achieve higher accuracy. Experimental results on the New York University Depth v2 (NYUDv2) dataset show effective results on the multi-level scene understanding.

- **spatial coarse-to-fine learning**: in this case, part-level categories (*e.g.*, *arm*) originate from the respective object level class (*e.g.*, *person*). This problem is defined in Section 6.3. The semantic segmentation of parts of objects in the wild is a challenging task in which multiple instances of objects and multiple parts within those objects must be detected in the scene. This problem remains nowadays very marginally explored, despite its fundamental importance towards detailed object understanding. We propose a novel framework combining higher object-level context conditioning and part-level spatial relationships to address the task. To tackle object-level ambiguity, a class-conditioning module is introduced to retain class-level semantics when learning part-level semantics. In this way, mid-level features carry also this information prior to the decoding stage. To tackle part-level ambiguity and localization we propose a novel adjacency graph-based module that aims at matching the relative spatial relationships between ground truth and predicted parts. The experimental evaluation on the Pascal-Part dataset shows that we achieve state-of-the-art results on this task.

6.2 Coarse-to-Fine Learning at Semantic Level

As we observed, we could interpret the scene at different levels of precision: in some scenarios, for example, it may be enough to identify just a few classes while in others a more fine-grained

prediction could be required. Moreover, in other settings, a coarse set of classes could be predicted first and then the set of classes could hierarchically grow into more refined categories to better understand the semantic context. To visualize this scenario, imagine an indoor navigation system first trained on a very coarse set of labels to segment, *e.g.*, *movable objects*, *permanent structures*, and *furniture*, in order to, *e.g.*, avoid obstacles. After a while, the dataset used for the initial training could be refined with a more fine-grained set of semantic classes (*e.g.*, the *movable objects* class could be split into *books*, *monitor*, *etc.*) and the task of the robotic system is to interact with these new types of objects. One solution could be to retrain from scratch the underlying neural network with the new set of classes; however, some other solutions may seem more reasonable. For instance, the initial prediction could be helpful for the learning process of the more refined set of classes in the form of an incremental learning approach where new tasks are accomplished at subsequent steps. Furthermore, solving multiple tasks at the same time could be beneficial in terms of both accuracy and the possibility to choose the appropriate set of labels for the particular task at hand (*e.g.*, object avoidance, object interaction, *etc.*).

Starting from these considerations, our main goal is to transfer previously gained knowledge, acquired on a simple semantic segmentation task with coarse classes (*e.g.*, *structures*, *furniture*, *objects*, *etc.*), to a new model where more fine-grained and detailed semantic classes (*e.g.*, *walls*, *beds*, *cups*, *etc.*) are introduced.

Notice that this task is similar to incremental (or continual) learning; however, there are a few key differences. With respect to incremental learning for semantic segmentation [18], here we do not add the ability to segment and label new classes; instead, we refine the initial prediction on a coarse set of classes with a more fine-grained set of classes originated from the previous one. In this sense, indeed, we expand the ability of the deep neural network to accomplish the new fine-grained task.

This type of learning paradigm moved its inspirations from the human way of learning concepts, and specifically from the way in which training examples are presented to the learner. Unlike most machine learning systems, humans do not learn new difficult tasks (*e.g.*, recognizing a rare variety of birds) entirely from scratch. Instead, new skills are often learned progressively, starting with easier tasks and gradually becoming able to tackle harder ones. For example, humans first learn to recognize birds before differentiating among the different species of birds [12, 165]. Thus, we can think of human learning as being driven by a curriculum that is either explicitly provided by a teacher, or implicitly learned. Hence, we can think of our works as a coarse-to-fine curriculum paradigm with only one intermediate stage of easier concepts towards the actual complex task.

The last focus of this work is the development of a new approach based on the simultaneous output of multi-level semantic maps. By exploiting domain sharing at the feature extraction level, the model will be fine-tuned to learn different levels of semantic labeling at the same time. This is somehow related to multitask learning [40, 166, 167], where multiple tasks are solved at the same time, while exploiting commonalities and differences across tasks. This can result in improved learning efficiency and prediction accuracy for the task-specific models, when compared to training the models separately. Training on multiple tasks could even bring additional information and, by sharing the representations between related tasks, we allow the model to generalize better on one or more tasks. Additionally, there is a reduction in the computational time with respect to training one independent architecture for each task. Our approach, however, is not strictly speaking multitask learning because we do not predict multiple different tasks at the same time (*e.g.*, semantic maps along with depth scene completion, surface normals prediction, *etc.*) but actually we learn multiple sets of classes at the same time allowing to have different levels of representations of the semantic map simultaneously, greatly reducing the time required for the training (a single training replaces multiple steps of training). Furthermore, we

share the same data across the tasks, which actually represent a different hierarchical clustering of the labels.

We based our approach on an end-to-end deep learning pipeline for semantic segmentation on color and depth data (*i.e.*, color representation with an associated depth information) based on the DeepLab-v3+ model [4]. To train the network and to evaluate the results, we employed the NYUDv2 dataset [168], which consists of a set of scenes of the indoor environment, with three sets of labels (5, 15 and 41 classes, respectively) at different levels of semantic precision and we compared our results with other recent methodologies, although not related to incremental or multitask approaches.

The remainder of this section is organized as follows: Section 6.2.1 presents an overview of related work; Section 6.2.2 introduces the proposed methodologies; the employed dataset and training procedures are described in Section 6.2.3; while the experimental results are discussed in Section 6.2.4.

6.2.1 Preliminaries

In this work, we also exploit depth data and we adapt approaches for color images to this scenario with some modifications at the earlier layers. Although this work focuses more on the incremental refinement than at achieving high performance on the stand-alone segmentation task, a brief overview of recent research papers exploiting both color and depth images is presented here. In [169, 170], a scheme involving CNN at multiple scales has been adopted. Two different CNNs for color and depth and a feature transformation network are exploited in [171]. In [172], a region splitting and merging algorithm for RGB-D data has been proposed. In [173], a MRF superpixel segmentation is combined with a tree-structured segmentation for scene labeling. Multiscale approaches have also been exploited (*e.g.*, [174]). Hierarchical segmentation based on the output of contour extraction has been used in [175], which also deals with object detection from the segmented data. Another combined approach for segmentation and object recognition has been presented in [168], which exploits an initial over-segmentation followed by a hierarchical scheme.

The last idea we want to investigate is the multiple learning of more representations at the same time, which is somehow related to multitask learning [40, 166, 167]. Multitask learning has been widely applied to semantic segmentation. For instance, in [170], a single multiscale CNN is employed to solve the three tasks of depth prediction, surface normals estimation, and semantic labeling. In [176], the semantic segmentation task is solved using three networks with shared features: differentiating instances, estimating masks, and categorizing objects. In [177], multi-task learning is employed to align the features computed from synthetic data while performing the predictions of the depth, the edges, and the surface normals with the ones computed from real-world images. In [178], the semantic segmentation map is predicted together with instance segmentation and depth prediction; additionally, an approach to select the weights for each loss is proposed.

6.2.2 Proposed Methods

In this section, we show in detail the approaches proposed in this work. Although the proposed procedures are agnostic to the underlying architecture, for the evaluation, we chose the DeepLab-v3+ [4, 5] network, which has state-of-the-art performance on segmentation tasks. The network consists of a Xception feature extractor, whose weights were pre-trained [179] on the Pascal VOC 2012 dataset [114], and a decoder made by Atrous Spatial Pyramid Pooling (ASPP)

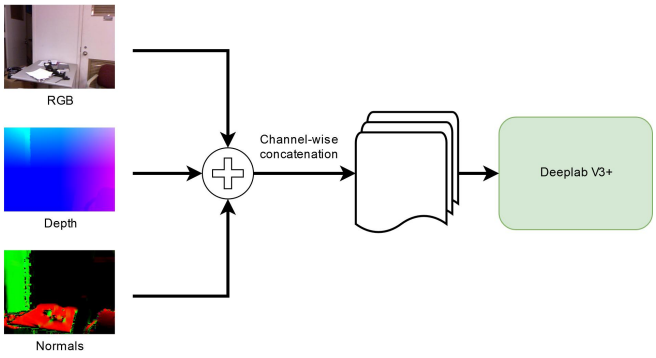


Figure 6.2: Early fusion of the different representations (color, depth, and surface normals).

layers. We evaluated our results on the NYUDv2 dataset after a pre-processing stage detailed in Section 6.2.3.

As with most deep learning approaches, the model takes as input a multi-channel tensor (in our case, nine channels corresponding to the color image, the depth information and the surface normals) and outputs the predicted *softmax* tensor with a number of channels equal to the number of predicted classes. This operation returns a probability distribution containing for each pixel the probability of belonging to each specific class. The class corresponding to the highest probability value is chosen for each pixel by an *argmax* operation. As a final result, we end up with the predicted segmentation map where each pixel value is the index of the class it belongs to.

To exploit the multiple types of information, we performed an early fusion of the different representations, *i.e.*, color, depth, and surface normals, as depicted in Figure 6.2, and then we feed them to the network. More in detail, each input tensor has nine channels. The first three channels correspond to the RGB color representation in the range $[-1, 1]$, *i.e.*, we divided by 127.5 the color values and then subtracted 1. The following three channels correspond to the geometry components where each channel represents the position of each pixel with respect to the three axes (X,Y,Z) of the 3D space. We normalized these values by subtracting the mean and dividing by the standard deviation along each axis. The last three channels represent the surface normal vectors. We used the standard representation with the three components of the unit vector perpendicular to the surface at each location (*i.e.*, the components assume values in the range $[-1, 1]$ and the vector norm is equal to 1).

For the training procedures, we employed the Jaccard loss (also known as Intersection over the Union (IoU) loss). We chose this loss since it has proven to be useful when training on a dataset with unbalanced numbers of pixels in the different classes within an image because it gives equal weight to all classes. Additionally, it has shown better perceptual quality than the usual cross-entropy loss with our setup in which there are some small objects and many under-represented labels in the dataset. The Jaccard loss is defined as:

$$\mathcal{L}_{Jaccard} = 1 - IoU = 1 - \frac{|\hat{y} \cap y|}{|\hat{y} \cup y|} \tag{6.1}$$

where $|\cdot|$ represents the cardinality of the considered set, $\hat{y}$ is the predicted segmentation map, and y is the ground-truth map.

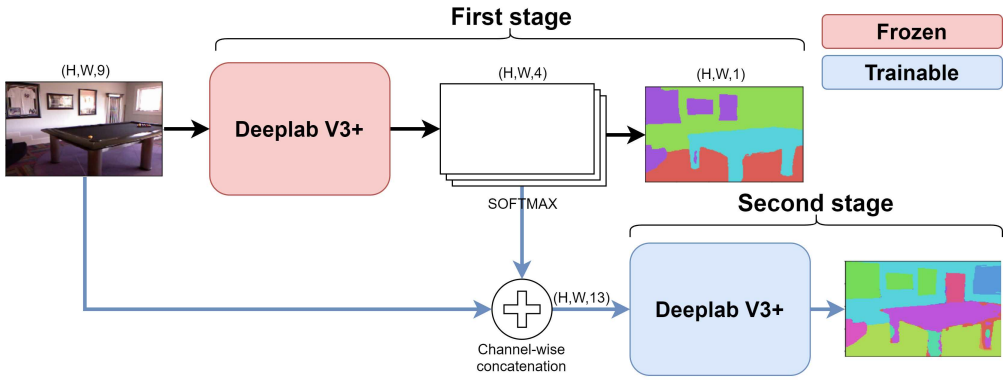


Figure 6.3: Diagram of the incremental approach where the softmax of the predictions at the first stage is concatenated as additional input for the second phase.

6.2.2.1 Hierarchical Learning

Here we present and discuss the various methods we designed for knowledge transfer in semantic segmentation.

In the first approach, we used a different DeepLab-v3+ model for each step (*i.e.*, on the considered dataset, we have a first model M_1 for the 5-class setting, a second M_2 for the 15-class setting, and a third M_3 for the 41-class setting). As every incremental learning approach, we start by training the M_1 DeepLab-v3+ model on the coarser set of classes (*e.g.*, five classes in our scenario). After that, we freeze the first model M_1 and we employ its output of the softmax operation as an additional input component when we train the model M_2 on the set of more fine-grained classes (*e.g.*, 15 classes). We repeat the same approach also when moving from M_2 to M_3 (*i.e.*, from 15 to 41 classes). This methodology was partially derived by the idea presented in [172] where the softmax information is used for binary classification task. Furthermore, notice that, when training for the finer tasks, the networks corresponding to the coarser ones are frozen, *i.e.*, we do not train in a single step a large size network containing the two (or three) networks for the two (or three) tasks but we perform a set of independent trainings each working on a single stage of the network.

More in detail, the number of predicted classes were 4, 13 and 40, respectively, because the *unknown* and the *unlabeled* classes were discarded as done by all competing approaches (see Section 6.2.3 for further details). Note that the number of trainable parameters remains constant during the two stages because in the incremental step the previous network is completely frozen and not trained anymore. The proposed framework is shown in Figure 6.3 and it is evident that the previous stage of training acts as a conditioning element for the following one. Indeed, the softmax tensor output from the first training stage (*i.e.*, from M_1) serves as additional input (concatenating it with the RGB images) for the second stage (model M_2) and the same idea is exploited when moving from M_2 to M_3. This way the network is constrained to learn the mapping from the coarser to the finer-grained sets of classes.

In the second approach, we fed as an additional input to the incremental stages the arg max of the predicted semantic map instead of the softmax. This approach is shown in Figure 6.4. The main difference from previous approach is that we only feed the index of the maximum of the predicted map and we drop the information about the probabilities of all the various classes which was before represented by the softmax vector for each pixel. In this way, we lose the information about the uncertainty of the prediction but, on the other side, the representation is

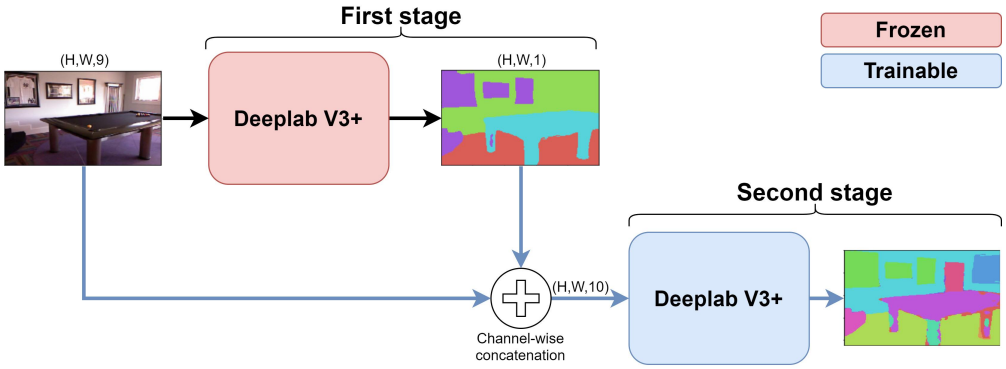


Figure 6.4: Diagram of the incremental approach where the argmax of the predictions at the first stage is concatenated as additional input for the second phase.

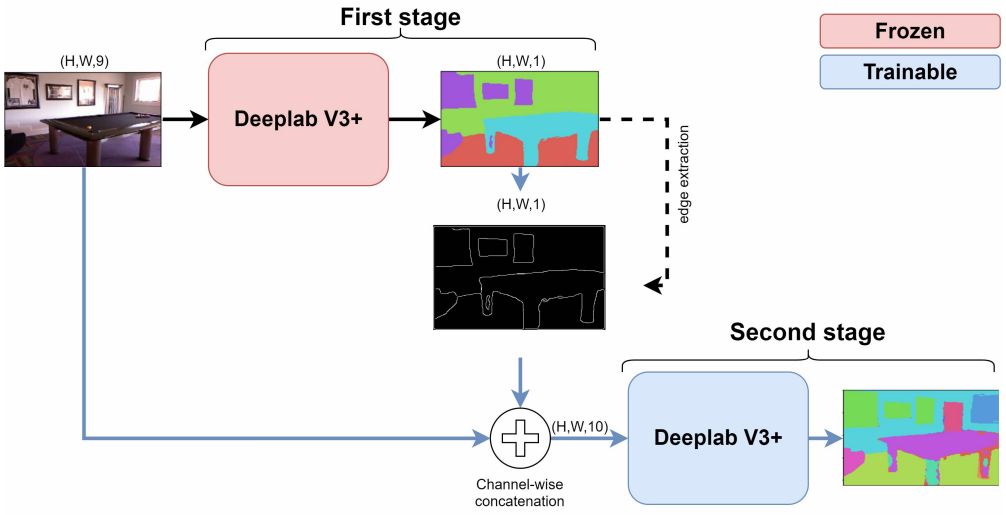


Figure 6.5: Diagram of the incremental approach where the edges of the predictions at the first stage are concatenated as additional input for the second phase.

much more compact having only a single value representing the predicted class for each pixel. Notice that the first approach (softmax) is more complex but leads to slightly better results (see the experimental evaluation in Section 6.2.4). On the other side, the second approach (argmax) is faster and simpler even if it has slightly worse performance.

In the third approach, we fed as additional input to the incremental stages the edges of the predicted semantic map. This approach is shown in Figure 6.5. Differently from before, the additional information channel does no longer contains the semantic labels of the classes but instead is represented by the boundary information. In this way, the second stage of training is more focused on the contours of the shapes, which are generally difficult to discriminate in semantic segmentation tasks.

We argue that combining multiple cues could lead to further improvements; however, this possibility is limited in practice by memory constraints of the employed GPUs.

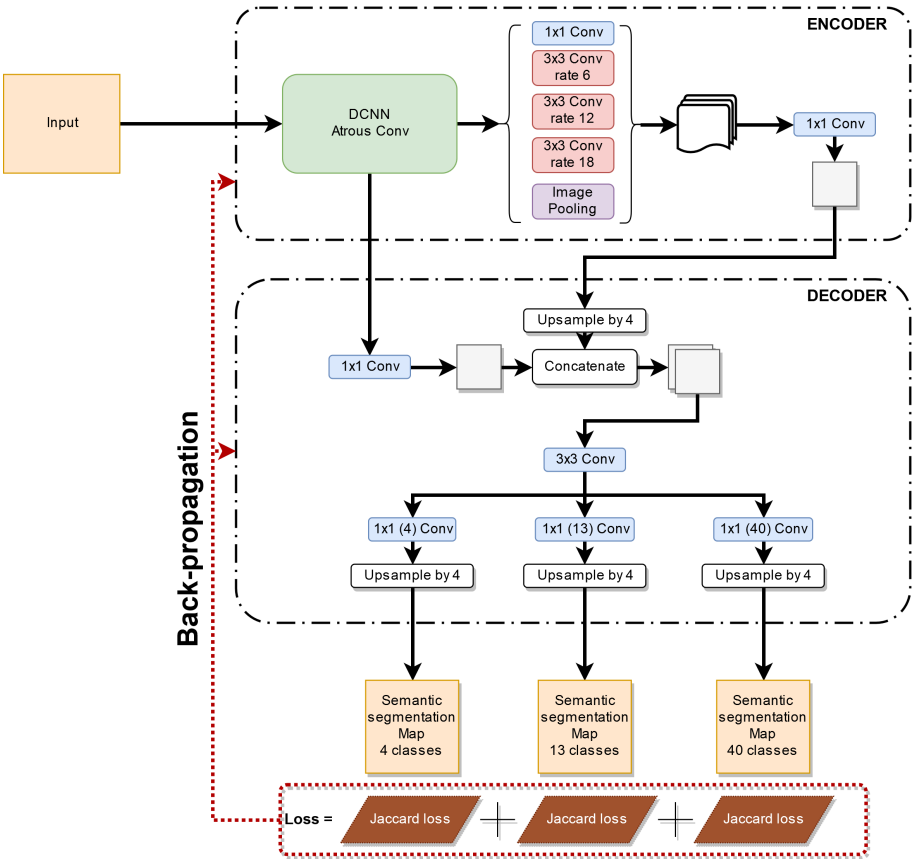


Figure 6.6: Modified DeepLab-v3+ architecture for joint learning of multiple representations.

6.2.2.2 Joint Learning of Multiple Representations

Finally, we started to investigate the prediction of different labelings at the same time and whether this could be helpful to improve performance on the coarser set of labels since we are learning more detailed information about their content and vice versa if the coarse labeling can help the fine one. We then designed a different decoder to accomplish the multiple representations and we trained the architecture end-to-end with three different losses (one for each set of labels). In this case, the complete loss function is defined as:

$$\mathcal{L}_{total} = \sum_{i=1,2,3} \lambda_i \mathcal{L}_{Jaccard,\mathcal{X}_i} \tag{6.2}$$

where $\mathcal{X}_i$ is the ith set of labels, *i.e.*, $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}_3$ contain, respectively, 5, 15, and 41 classes, while the hyper-parameters λ_i balance the three losses. These were empirically set to 1 so that all the terms contribute equally during the back-propagation phase. The loss associated with the set $\mathcal{X}_i$ is then written as $\mathcal{L}_{Jaccard,\mathcal{X}_i}$. The approach is illustrated in Figure 6.6: we used a single standard DeepLab-v3+ encoder while the decoder has been modified to be able to deal with the multiple tasks together. From this figure, we can appreciate that the first part of the decoder is shared across all the tasks while the last 1×1 convolution layer is unique for each segmentation

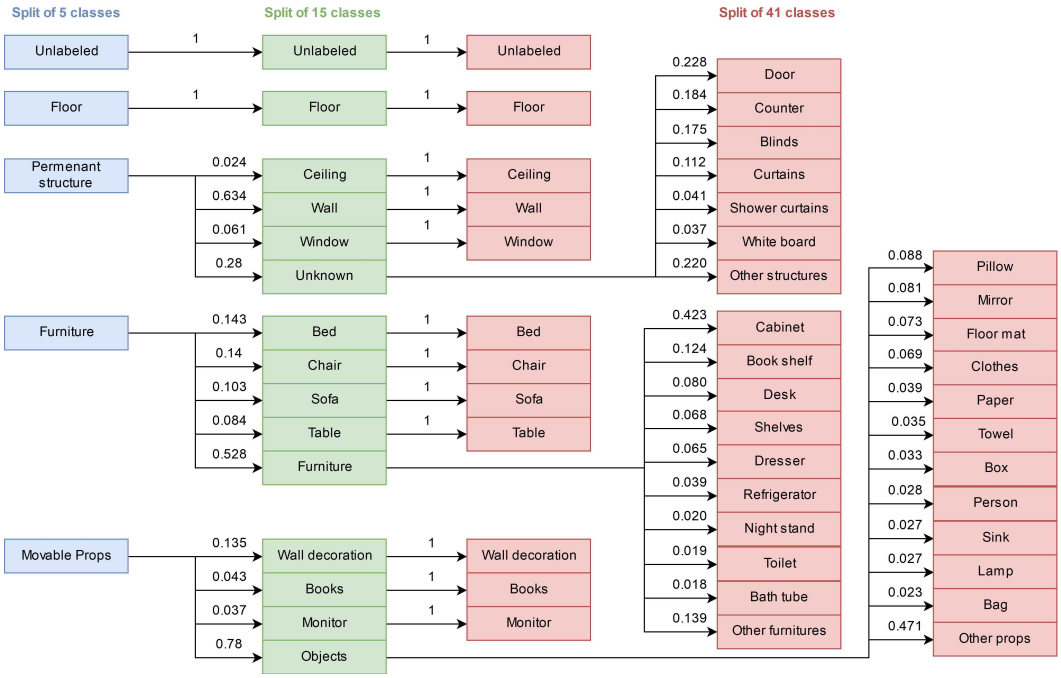


Figure 6.7: Diagram showing the hierarchical mapping between the three different set of classes (blue for the split of 5, green for the split of 15, red for the split of 41). The numbers above the arrows are the fraction of the parent class that is assigned to each of its derived ones.

task (*i.e.*, 4, 13, and 40 classes segmentation in our case, after excluding the *unknown* and the *unlabeled* classes as detailed in Section 6.2.3). The final 1×1 layers are followed by a bilinear upsampling procedure to restore the original input dimensions and a softmax classification layer is then applied to each output to get the final predictions.

6.2.3 Training on the NYUDv2 Dataset

The NYUDv2 dataset [168] was used to train the proposed architectures and to evaluate the performance of the proposed approach. This dataset contains 1449 depth maps and color images of indoor scenes acquired with a first generation Kinect sensor divided into a training set with 795 scenes and a test set with the remaining 654 scenes. The original resolution of the images is 640×480; however, for the training procedures, we employed a lower resolution of 560×425 for memory constraints. The evaluation of the results, instead, was carried out on the original resolution images for a fair comparison with competing approaches. For results evaluation, we used the ground truth labels from [180], and we considered the three clusters of 5, 15, and 41 labels, respectively, as mapped in [168, 181].

In particular, the three considered set of labels are hierarchically represented in Figure 6.7 where we can appreciate how the derived classes emerge from parent ones. Two classes, *i.e.*, *unlabeled* and *floor*, are peculiar because they are never split when moving to finer semantic representations. Similarly, various classes in the set of 15 labels are not split when moving to the finer set of 41. From the diagram, we can notice how there are clear unbalanced splitting situations. For instance, some classes of the split of 15 are underrepresented in the dataset, as

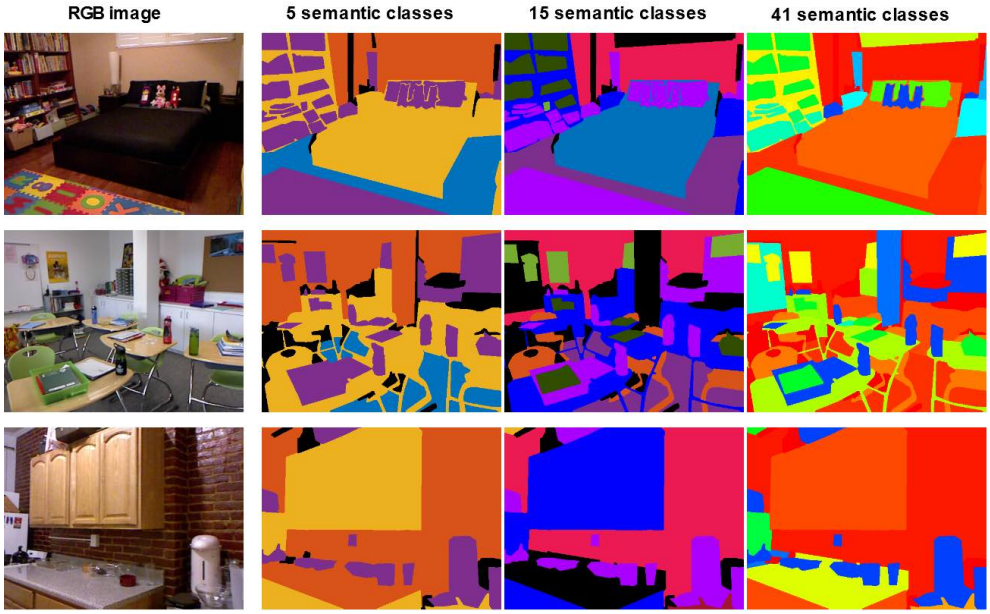


Figure 6.8: Sample scenes from the NYUDv2 dataset highlighting the different levels of semantic description in the segmentation maps. From left to right: RGB image, semantic map with 5 classes, semantic map with 15 classes, and semantic map with 41 classes.

can be appreciated from the very low fraction of pixels in these classes from parent ones: *e.g.*, *ceiling* and *window* are present in only 2.4% and 6.1%, respectively, of instances of *permanent structure*; and *monitor* and *books* are derived only in the 3.7% and 4.3% of the *movable props* parent class. Additionally, the splitting is not uniformly distributed among the parent classes, thus from 3 out of 5 classes of the split of 5 derive 13 out of 15 classes of the split of 15. If we move to the analysis of the split of 41, the considerations become even more severe. There are few classes deriving from $\leq 2\%$ of instances of parent class, *e.g.*, *bath tub*, *toilet*, *night stand*, and many others (10 classes) deriving from 2–5%. Moreover, it should be noticed that, in this case, 29 classes out of 41 derive from just three parent classes, thus confirming the extreme inhomogeneity of this splitting.

As done by all the competing approaches (*e.g.*, [170, 172, 181, 182]), we removed both from the prediction and from the evaluation of the results the *unlabeled* and *unknown* classes when present. Indeed, they are fictitious classes artificially created during the labeling procedures of the images. This choice allowed directly comparing the results with competing approaches, although not related to incremental or multi-tasking learning.

Moreover, in Figure 6.8, we can appreciate the various level of semantic understanding which have been considered for the evaluation. For instance, in the first row, we can visualize how the generic *furniture* class in the set of five classes (in yellow) is split into *bed* and *furniture* in the set of 15 classes (light blue and blue, respectively) and that *bed* is further refined into *bed* and *pillows* in the set of 41 classes (in orange and light green, respectively). Again, in the second row, for example, we can appreciate how the generic class *movable props* of the set of five classes (in purple) is then refined to *books* and *object* in the set of 15 classes (in dark green and light purple, respectively), and then further refined in the last set of classes. Finally, in the third row, we can visualize how the *permanent structure* class in the set of five classes (in orange) is then

split into the classes *ceiling* and *wall* in the set of 15 classes (in yellow and pink, respectively).

The various approaches were trained on the NYUDv2 training set using the three different sets of labels. We employed Stochastic Gradient Descent (SGD) and ran the procedure for 100 epochs. The initial learning rate lr_0 was set to $lr_0 = 10^{-2}$, the weight decay w_d to $w_d = 0.9$, and the batch size equal to 2. The learning rate scheduler decreased the learning rate lr every $s = 2$ epochs using the following formula:

$$lr(ep) = lr_0 \cdot w_d \cdot \exp \lfloor \frac{ep}{s} \rfloor \tag{6.3}$$

where ep denotes the index of the current epoch.

We used TensorFlow [125] to develop and train our framework. For each stage, the number of trainable parameters and FLOPs was roughly the same as the original DeepLab-v3+ architecture, *i.e.*, 41M and 82B, respectively (the added components require a very small number of parameters with respect to the DeepLab model). The training of the neural network took about 22 hours on a NVIDIA Tesla K40 GPU with Intel(R) Core(TM) i7 CPU 970 @ 3.2 Ghz. The implementation of the proposed model is available at `https://github.com/LTTM/IL-Coarse2Fine`.

6.2.4 Experimental Results

In this section, we discuss the performance of the various proposed approaches in the two different settings of incremental and multi-task learning.

First, we start by comparing our modified DeepLab-v3+ architecture with early fusion of the three information representations, *i.e.*, color, depth, and surface normals, with some recent works. To evaluate the results, we employed the most widely used metrics for semantic segmentation problems: the Pixel Accuracy (PA), the mean Class Accuracy (mCA), and the mean Intersection over Union (mIoU) [124].

The modified network is able to obtain state-of-the-art results on all the three set of labels. In Table 6.1, we can confirm that our baseline network could outperform competing approaches in terms of both PA and mCA on the split of four classes; additionally, we also show the obtained mIoU. Similar results were achieved by our baseline model for the set of 13 classes, as shown in Table 6.2, while on the set of 40 labels some very recent approaches have better performance (see Table 6.3). However, notice that the aim of this work is to propose an efficient hierarchical learning strategy, not to improve the performance on the segmentation task by itself.

Then, we evaluated our hierarchical learning approaches. Firstly, we started from the coarser set of four classes and we moved to the prediction of 13 classes: the results are shown as the last three lines of Table 6.2 for the three different approaches. In this case, we can appreciate that the addition of the softmax information from the four-class model or of the edges information are useful cues to reach higher accuracy on the new set of classes if compared with the baseline counterpart. In particular, in the case of softmax or edges information, there are improvements in all three considered metrics with respect to the baseline DeepLab-v3+. In particular, the softmax information leads to the best class accuracy (almost 70%) while the use of edge information is the best strategy with respect to the pixel accuracy (76.3%) and to the mIoU (53.9%). Notice that the mIoU gap with respect to the direct training on the 13 classes is about 2.5% (by the way, this metric and the mCA are more interesting for our task since the pixel accuracy is strongly dependent on large structures such as the floor that are not split in the hierarchical labeling). The argmax information, instead, brings a limited contribution to the final accuracy values.

One may wonder what would happen if we train both the first and the second stage with the same set of classes (*i.e.*, by just using a deeper network without really exploiting the hierarchical structure of the classes). We expect this scenario to achieve almost the same results of our

Table 6.1: Semantic segmentation performances on the NYUDv2 dataset with four classes of the proposed method and of some competing approaches (the table shows percentage values). We underlined the best result among all the methods for each metric, while the best result among the proposed techniques is reported in bold.

Method	**PA**	**mCA**	**mIoU**
Silberman *et al.* [168]	59.6	58.6	-
Ren *et al.* [173]	73.0	58.0	-
Măller *et al.* [183]	71.9	72.3	-
Gupta *et al.* [184]	78.0	64.0	-
Cadena *et al.* [185]	66.9	65.2	-
Stuckler *et al.* [186]	70.9	65.0	-
Couprie *et al.* [181]	64.5	63.5	-
Eigen *et al.* [170]	83.2	82.0	-
DeepLab-v3+	**82.5**	**82.2**	**70.3**

Table 6.2: Semantic segmentation performances on the NYUDv2 dataset with 13 classes of the proposed methods and of some competing approaches (the table shows percentage values). We underline the best result among all the methods for each metric, while the best result among the proposed techniques is reported in bold.

Method	**PA**	**mCA**	**mIoU**
Wang *et al.* [187]	-	42.2	-
Hermans *et al.* [188]	54.2	48.0	-
Khan *et al.* [189]	58.3	45.1	-
Couprie *et al.* [181]	52.4	36.2	-
Pagnutti *et al.* [182]	67.2	54.4	-
Michieli *et al.* [172]	67.2	54.5	-
Eigen *et al.* [170]	75.4	66.9	-
Baseline (DeepLab-v3+)	75.5	68.2	51.4
Stacking (2 concatenated DeepLab-v3+)	75.7	68.9	51.9
Incremental (softmax)	76.1	**69.9**	52.9
Incremental (argmax)	74.8	68.3	51.3
Incremental (edges)	**76.3**	69.8	**53.9**

baseline approach, or slightly higher, since we are retraining the same architecture with an additional input, which is the output of the previously trained network. At the same time, we expect that the incremental framework is the dominant factor for the performance increase. Indeed, the results for this stacking are perfectly in line with our intuition, as reported in Table 6.2 with the name "Stacking".

Furthermore, we performed an additional incremental step to predict the set of 40 classes starting from the prediction of the set of 13 labels. The results are reported in Table 6.3. In this case, our method was outperformed by some methods in the literature, due to some inner limitations of the employed DeepLab-v3+ architecture. However, the most interesting thing for this work is the comparison with our baseline method, *i.e.*, DeepLab-v3+ directly trained on the 40 classes, in order to appreciate the gain of the hierarchical approaches.

We can appreciate how the three additional cues employed produce some improvements in the various metrics even if the gain is more limited. The result is still noticeable if we remember

Table 6.3: Semantic segmentation performance on the NYUDv2 dataset with 40 classes of the proposed methods and of some competing approaches (the table shows percentage values). We underline the best result among all the methods for each metric, while the best result among the proposed techniques is reported in bold.

Method	**PA**	**mCA**	**mIoU**
Silberman *et al.* [168]	54.6	19.0	-
Ren *et al.* [173]	49.3	21.1	21.4
lin2017cascaded *et al.* [190]	-	-	<u>47.7</u>
Wang *et al.* [187]	-	47.3	-
Gupta *et al.* [184]	60.3	35.1	31.3
Long *et al.* [1]	66.9	<u>65.2</u>	-
Liu *et al.* [191]	<u>70.3</u>	51.7	41.2
Qi *et al.* [192]	-	55.7	43.1
Eigen *et al.* [170]	65.6	45.1	34.1
Baseline (DeepLab-v3+)	60.0	33.3	22.0
Incremental (softmax)	59.1	33.5	**22.1**
Incremental (argmax)	**61.3**	30.7	20.7
Incremental (edges)	59.2	**34.0**	**22.1**

Table 6.4: Experimental results on NYUDv2 with simultaneous output of the three segmentation maps, percentage values. The best results are highlighted in bold.

Method	**4 Classes**			**13 Classes**			**40 Classes**		
	PA	**mCA**	**mIoU**	**PA**	**mCA**	**mIoU**	**PA**	**mCA**	**mIoU**
DeepLab-v3+	82.5	82.2	70.3	75.5	**68.2**	51.4	60.0	33.3	22.0
Multi-tasking	**83.2**	**82.3**	**72.0**	**75.6**	**68.2**	**51.7**	**61.0**	**33.3**	**22.1**

that the splitting is highly unbalanced and inhomogeneous as we have seen in Section 6.2.3.

Finally, in Table 6.4, we evaluate our joint learning approach on the three sets of classes simultaneously. We can appreciate that the joint model allows not only to predict the three sets of labels at the same time, without the need for multiple training stages, but also to improve the accuracy with respect to the baseline in all the scenarios and for all the metrics. The improvement, although consistent across all experiments and metrics, is sometimes modest and smaller than some of the previously proposed methods. The highest gains are achieved in the PA and mIoU for the set of four classes and in the PA for the set of 40 classes. It should be noticed that the chosen architecture is already highly performing, especially on the sets of 4 and 13 classes.

Figure 6.9 shows some qualitative results for the set of five classes. Here, we can compare the performance of our baseline DeepLab-v3+ with respect to the multi-tasking approach (the other methods do not apply to this setting since there is no coarser representation). As already noticed, both the approaches have very high accuracy on this task. The image in the first column is very similar between the two approaches; however, we can verify how the multi-task learning outperforms the baseline in the remaining four images. In particular, look at the purple object on the top left of the figure in second column, the orange top of the *furniture* on the left, or again to the purple objects on the center-right of the image. In column 3, we can clearly see an improvement in the definition of the shapes of the sofa and of the pillows. Similarly, the *wall* and the *furniture* are better recognized in the last two images.

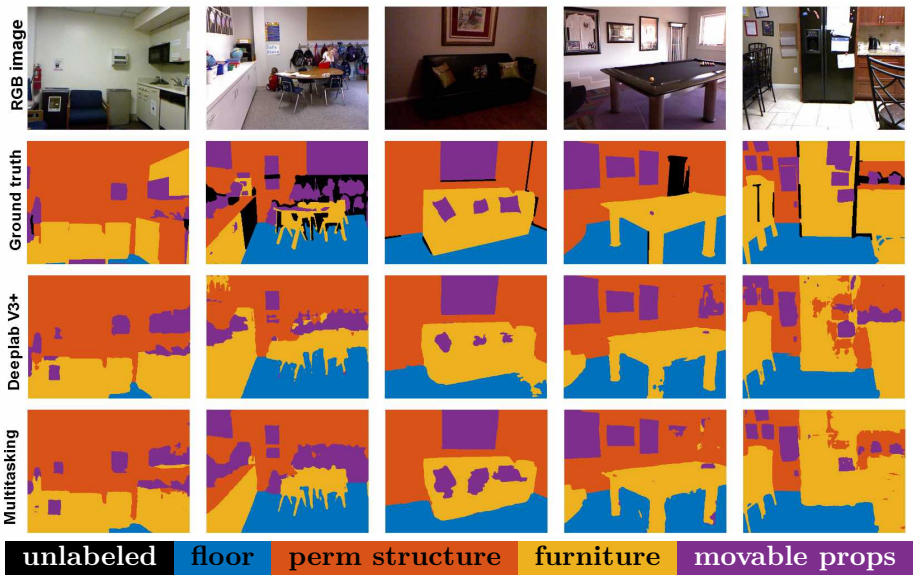


Figure 6.9: Qualitative results for the set of 5 classes of the proposed approaches. From left to right, the chosen images are the ones numbered as: 0, 124, 145, 168, and 368.

Finally, we report in Figure 6.10 the qualitative results for the split of 15 classes for all the proposed approaches. From the figure, we can appreciate that the incremental approach with the softmax or the edges generally lead to a much cleaner prediction with few artifacts. This can be seen particularly well in column 1, where the *chair* in the center of the image is fully recovered by the proposed incremental methods, and in column 3, where the *wall* is cleaned from prediction errors (in this scene, the *sofa* has also been properly segmented but a wrong label of *bed* has been associated to it). In the same scenes, the baseline, the incremental approach with the argmax, and the multi-tasking suffer from some artifacts. In column 4, we can notice that the edges information revealed to be more significant than the softmax information to properly recognize the leg of the pool table.

In general, we argue that the incremental approaches with edges or softmax information are much more reliable than the conditioning based on the argmax: in the softmax case, a larger amount of information is fed as additional input giving the network the possibility to discriminate between certain or uncertain predictions while in the edges case the network is forced to focus more on the edges of the objects, which typically represent one of the most challenging characteristics to be learned.

For what concern computational requirements, the inference time of the modified DeepLab-v3+ network is 23 ms on the workstation used for the training (with an Intel 970 CPU and a Nvidia K40 GPU), which is roughly the same as the standard DeepLab-v3+ architecture. Notice that incremental schemes require multiple inferences, *e.g.*, the 13 classes experiment requires executing the model two times in cascade (one for the four-class network and one for the 13-class network taking in input also the outcome of the four-class model). If real-time performance were needed, the best option would be the joint learning scheme proposed in Section 6.2.2.2 that is able to perform all three tasks with a single pass through the encoder module that is the most computationally demanding stage being the decoder very lightweight.

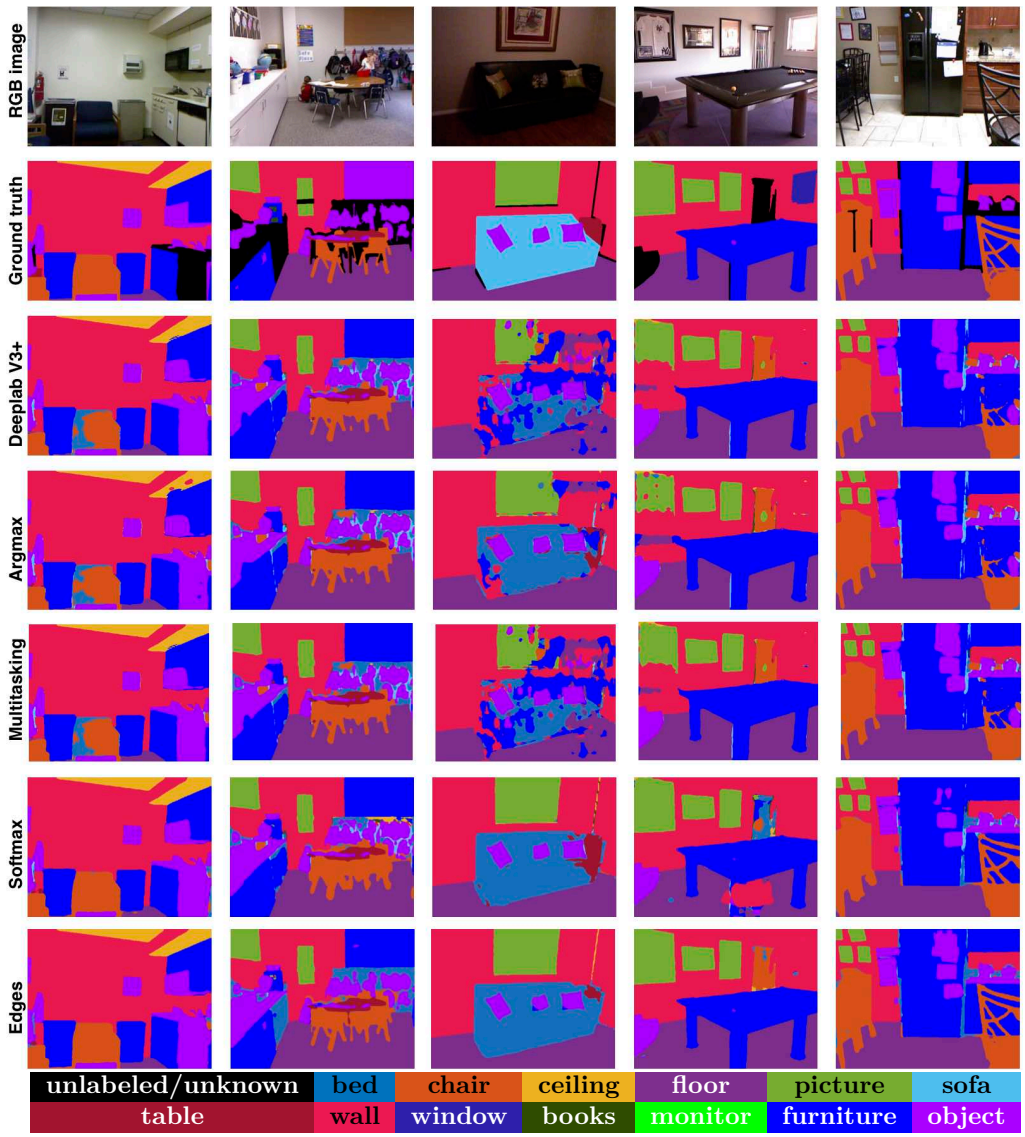


Figure 6.10: Qualitative results for the set of 15 classes of the proposed approaches. From left to right, the chosen images are the ones numbered as: 0, 124, 145, 168, 368.

6.3 Coarse-to-Fine Learning at the Spatial Level

The segmentation and labeling of parts of objects can be regarded as a special case of semantic segmentation that focuses on parts decomposition. The information about parts provides a richer representation useful for many fine-grained tasks, such as pose estimation [193, 194], category detection [195–197], fine-grained action detection [198] and image classification [199, 200]. However, current approaches for semantic segmentation are not optimized to distinguish between different semantic parts since corresponding parts in different semantic classes often share similar appearance. Additionally, they only capture limited local context while the precise localization of semantic part layouts and their interactions requires a wider perspective of the image. Thus, it is not sufficient to take standard semantic segmentation methods and treat each part as an independent class. In the literature, object-level semantic segmentation has been extensively studied. Part parsing, instead, has only been marginally explored in the context of a few specific single-class objects, such as humans [201–204], cars [205, 206] and animals [207–209]. Multi-class part-based semantic segmentation has only been considered in a recent work [210], due to the challenging scenario of part-level as well as object-level ambiguities. Here, we introduce an approach dealing with the semantic segmentation of an even larger set of parts and we demonstrate that the proposed methodology is able to deal with a large amount of parts contained in the scenes.

Nowadays, one of the most active research directions is the transfer of previous knowledge, acquired on a different but related task, to a new situation. Different interpretations may exist to this regard. In the class-incremental task (see previous chapters), the learned model is updated to perform a new task whilst preserving previous capabilities: many methods have been proposed for image classification [48,49,81], object detection [51] and semantic segmentation [18,20–23,56]. Another aspect regards the coarse-to-fine refinement at the *semantic level*, in which previous knowledge acquired on a coarser task is exploited to perform a finer task [24, 211, 212] (see previous section). In this section, instead, we investigate the coarse-to-fine refinement at the *spatial level*, in which object-level classes are split into their respective parts [208, 210, 213].

More precisely, we investigate the multi-object and multi-part parsing in the wild, which simultaneously handles all semantic objects and parts within each object in the scene. Even strong recent baselines for semantic segmentation, such as FCN [1], SegNet [214], PSPNet [3] or DeepLab [5, 6], face additional challenges when dealing with this task, as shown in [210]. In particular, the simultaneous appearance of multiple objects and the inter-class ambiguity may cause inaccurate boundary localization and severe classification errors. For instance, animals often have homogeneous appearance due to furs on the whole body. Additionally, the appearance of some parts over multiple object classes may be very similar, such as cow legs and sheep legs. Current algorithms heavily suffer from these aspects. To address object-level ambiguity, we propose an object-level conditioning to serve as guidance for part parsing within the object. An auxiliary reconstruction module from parts to objects further penalize predictions of parts in regions occupied by an object which does not contain the predicted parts within it. At the same time, to tackle part-level ambiguity, we introduce a graph-matching module to preserve the relative spatial relationships between ground truth and predicted parts.

When people look at scenes, they tend to locate first the objects and then to refine them via semantic part parsing [213]. This is the same rationale for our class-conditioning approach, which consists of an approach to refine parts localization exploiting previous knowledge. In particular, the object-level predictions of the model serve as a conditioning term for the decoding stage on the part-level. The predictions are processed via an object-level semantic embedding Convolutional Neural Network (CNN) and its features are concatenated with the ones produced by the encoder of the part-level segmentation network. The extracted features are enriched with

this type of information prior, guiding the output of the part-level decoding stage. We further propose to address part-level ambiguity via a novel graph-matching technique applied to the segmentation maps. A couple of adjacency graphs are built from neighboring parts both from the ground-truth and from the predicted segmentation maps. Such graphs are weighted with the normalized number of adjacent pixels to represent the strength of connection between the parts. Then, a novel loss function is designed to enforce their similarity. These provisions allow the architecture to discover the differences in appearance between different parts within a single object, and at the same time to avoid the ambiguity across similar object categories.

The main contributions of this work can be summarized as follows:

1. We tackle the challenging multi-class part parsing via an object-level semantic embedding network conditioning the part-level decoding stage.

2. We introduce a novel graph-matching module to guide the learning process toward accurate relative localization of semantic parts.

3. Our approach (GMNet) achieves new state-of-the-art performance on multi-object part parsing on the Pascal-Part dataset [195]. Moreover, it scales well to large sets of parts.

6.3.1 Preliminaries

Single-Object Part Parsing has been actively investigated in the recent literature. However, most previous work assumes images containing only the considered object, well-localized beforehand and with no occlusions. Single-object parts parsing has been applied to animals [207], cars [204–206] and humans parsing [201–204]. Traditional deep neural network architectures may also be applied to part parsing regarding each semantic part as a separate class label. However, such strategies suffer from the high similar appearance between parts and from large scale variations of objects and parts. Some coarse-to-fine strategies have been proposed to tackle this issue. Hariharan *et al.* [211] propose to sequentially perform object detection, object segmentation and part segmentation with different architectures. However, there are some limitations, in particular the complexity of the training and the error propagation throughout the pipeline. An upgraded version of the framework has been presented in [213], where the same structure is employed for the three networks and an automatic adaptation to the size of the object is introduced. In [208] a two-channels FCN is employed to jointly infer object and part segmentation for animals. However, it uses only a single-scale network not capturing small parts and a fully connected CRF is used as post-processing technique to explore the relationship between parts and body to perform the final prediction. In [215] an attention mechanism that learns to softly weight the multi-scale features at each pixel location is proposed.

Some approaches resort to structure-based methodologies, *e.g.*, compositional, to model part relations [207, 208, 216–219]. Wang *et al.* [207] propose a model to learn a mixture of compositional models under various poses and viewpoints for certain animal classes. In [216] a self-supervised structure-sensitive learning approach is proposed to constrain human pose structures into parsing results. In [217, 218] graph LSTMs are employed to refine the parsing results of initial over-segmented superpixel maps. Pose estimation is also useful for part parsing task [212, 216, 219–221]. In [212], the authors refine the segmentation maps by supervised pose estimation. In [220] a mutual learning model is built for pose estimation and part segmentation. In [219], the authors exploit anatomical similarity among humans to transfer the parsing results of a person to another person with similar pose. In [221] multi-scale features aggregation at each pixel is combined with a self-supervised joint loss to further improve the feature discriminative capacity. Other approaches utilize tree-based approach to hierarchically partition the

parts [206, 222]. Lu *et al.* [206] propose a method based on tree-structured graphical models from different viewpoints combined with segment appearance consistency for part parsing. Xia *et al.* [222] firstly generate part segment proposals and then infer the best ensemble of parts-segment through and-or graphs. Knowledge graphs have been used for compositional object classification based on object parts in [223, 224].

Even though single-object part parsing has been extensively studied so far, **Multi-Object and Multi-Part Parsing** has been considered only recently [210]. In this setup, most previous techniques fail struggling with objects that were not previously well-localized, isolated and with no occlusions. Zhao *et al.* in [210] tackle this task via a joint parsing framework with boundary and semantic awareness for enhanced part localization and object-level guidance. Part boundaries are detected at early stages of feature extraction and then used in an attention mechanism to emphasize the features along the boundaries at the decoding stage. An additional attention module is employed to perform channel selection and is supervised by a supplementary branch predicting the semantic object classes.

6.3.2 Proposed Method

When we look at images, we often firstly locate the objects and then the more detailed task of semantic part parsing is addressed using mainly two priors: (1) object-level information and (2) relative spatial relationships among parts. Following this rationale, the semantic parts parsing is supported by the information coming from an initial prediction of the coarse object-level set of classes and by a graph-matching strategy working at the part-level.

An overview of our framework is shown in Figure 6.11. We employ two semantic segmentation networks $\mathcal{A}_o$ and $\mathcal{A}_p$ trained for the objects-level and part-level task respectively, together with a semantic embedding network $\mathcal{S}$ transferring and processing the information of the first network to the second to address the object-level prior. This novel coarse-to-fine strategy to gain insights into parts detection will be the subject of this section. Furthermore, we account for the second prior exploiting an adjacency graph structure to mimic the spatial relationship between neighboring parts to allow for a general overview of the semantic parts as described in Section 6.3.3.

The semantic segmentation networks have an autoencoder structure and can be written as the composition of an encoder and a decoder as $\mathcal{A}_o = \{\mathcal{E}_o, \mathcal{D}_o\}$ and $\mathcal{A}_p = \{\mathcal{E}_p, \mathcal{D}_p\}$ for the object-level and part-level networks, respectively. We employ the DeepLab-v3 [6] segmentation network with Resnet-101 [152] as encoder. The network $\mathcal{A}_o$ is trained using the object-level ground truth labels and then kept fixed. It extracts object-level segmentation maps which serve as a guidance for the decoder of the part-level network $\mathcal{D}_p$, in order to avoid the ambiguity across similar object categories. We achieve this behavior by feeding the output maps of $\mathcal{A}_o$ to an object-level semantic embedding network. In this work, we used a CNN (denoted with $\mathcal{S}$) formed by a cascade of 4 convolutional layers with stride of 2, square kernel sizes of 7, 5, 3, 3, and channel sizes of 128, 256, 512, 1024.

The part-level semantic segmentation network $\mathcal{A}_p$ has the same encoder architecture of $\mathcal{A}_o$. Its decoder $\mathcal{D}_p$, instead, merges the features computed on the RGB image and the ones computed on the object-level predicted map via multiple channel-wise concatenations. More in detail, each layer of the decoder considers a different resolution and its feature maps are concatenated with the layer at corresponding resolution of $\mathcal{S}$. In this way, the combination is performed at multiple resolutions in the feature space to achieve higher scale invariance as shown in Figure 6.11.

Formally, given an input RGB image $\mathbf{X} \in \mathbb{R}^{W \times H}$, the concatenation between part and object-

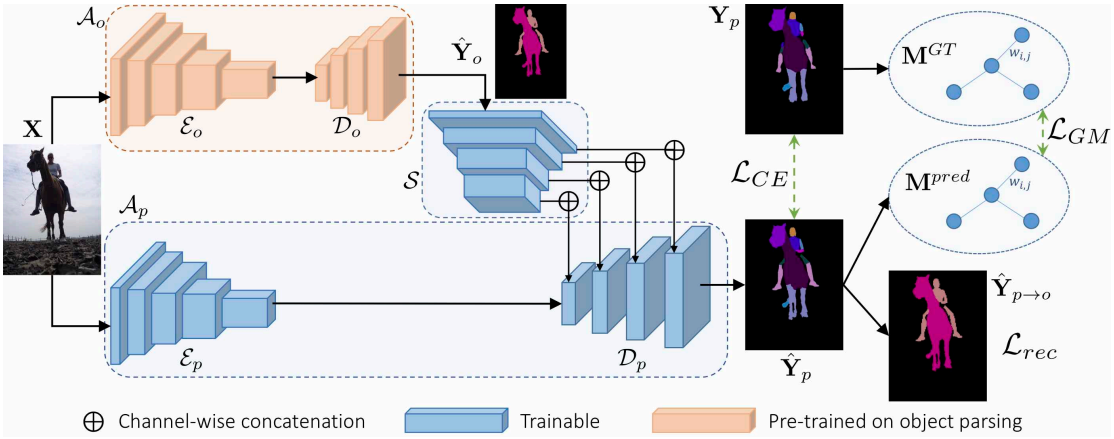


Figure 6.11: Architecture of the proposed Graph Matching Network (GMNet) approach. A semantic embedding network takes as input the object-level segmentation map and acts as high level conditioning when learning the semantic segmentation of parts. On the right, a reconstruction loss function rearranges parts into objects and the graph matching module aligns the relative spatial relationships between ground truth and predicted parts.

level aware features is formulated as:

$$\mathcal{F}_i(\mathbf{X}) = \mathcal{D}_{p,i}(\mathbf{X}) \oplus \mathcal{S}_{k+1-i}(\mathcal{A}_o(\mathbf{X})) \qquad i = 1, ..., k \tag{6.4}$$

where $\mathcal{D}_{p,i}$ is the i-th decoding layer of the part segmentation network, $\mathcal{S}_i$ denotes the i-th layer of $\mathcal{S}$, k is the number of layers and matches the number of upsampling stages of the decoder (*e.g.*, $k = 4$ in the DeepLab-v3), $\mathcal{F}_i$ is the input of $\mathcal{D}_{p,i+1}$. Since the object-level segmentation is not perfect, in principle, errors from the predicted class in the object segmentation may propagate to the parts. To account for this, similarly to [213], here we do not make premature decisions but the channel-wise concatenation still leaves the final decision of the labeling task to the decoder.

The training of the proposed framework (*i.e.*, of $\mathcal{A}_p$ and $\mathcal{S}$, while $\mathcal{A}_o$ is kept fixed after the initial training) is driven by multiple loss components. The first is a standard cross-entropy loss $\mathcal{L}_{CE}$ to learn the semantic segmentation of parts:

$$\mathcal{L}_{CE} = \sum_{c_p=1}^{N_p} \mathbf{Y}_p[c_p] \cdot \log\left(\hat{\mathbf{Y}}_p[c_p]\right) \tag{6.5}$$

where $\mathbf{Y}_p$ is the one-hot encoded ground truth map, $\hat{\mathbf{Y}}_p$ is the predicted map, c_p is the part-class index and N_p is the number of parts.

The object-level semantic embedding network is further guided by a reconstruction module that rearranges parts into objects. This is done applying a cross-entropy loss between object-level one-hot encoded ground truth maps $\mathbf{Y}_o$ and the summed probability $\hat{\mathbf{Y}}_{p\to o}$ derived from the part-level prediction. More formally, defining l as the parts-to-objects mapping such that object j contains parts from index $l[j-1]+1$ to $l[j]$, we can write the summed probability as:

$$\hat{\mathbf{Y}}_{p\to o}[j] = \sum_{i=l[j-1]+1,...,l[j]} \hat{\mathbf{Y}}_p[i] \qquad j = 1, ..., N_o \tag{6.6}$$

where N_o is the number of object-level classes and $l[0] = 0$. Then, we define the reconstruction

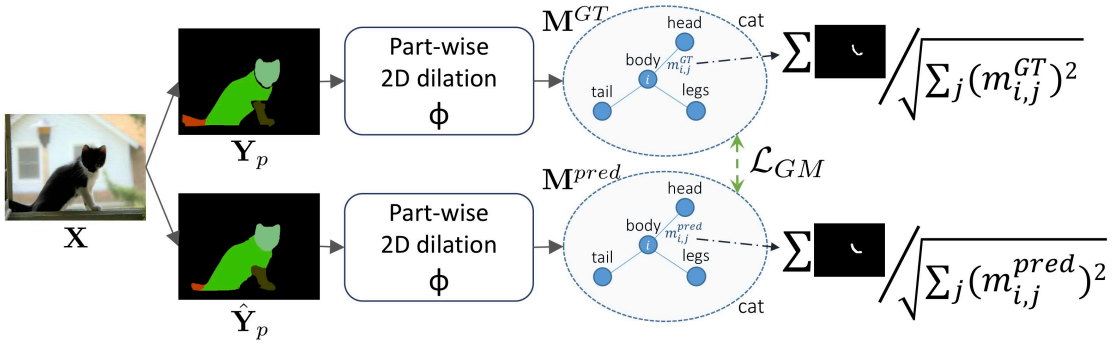


Figure 6.12: Overview of the graph matching module. In this case, cat's head and body would be considered as detached without the proper morphological dilation over the parts.

loss as:

$$\mathcal{L}_{rec} = \sum_{c_o=1}^{N_o} \mathbf{Y}_o[c_o] \cdot \log\left(\hat{\mathbf{Y}}_{p\to o}[c_o]\right) \tag{6.7}$$

The auxiliary reconstruction function $\mathcal{L}_{rec}$ acts differently from the usual cross-entropy loss on the parts $\mathcal{L}_{CE}$. While $\mathcal{L}_{CE}$ penalizes wrong predictions of parts in all the portions of the image, $\mathcal{L}_{rec}$ only penalizes for part-level predictions located outside the respective object-level class. In other words, the event of predicting parts outside the respective object-level class is penalized by both the losses. Instead, parts predicted within the object class are penalized only by $\mathcal{L}_{CE}$, *i.e.*, they are considered as a less severe type of error since, in this case, parts only need to be properly localized inside the object-level class.

6.3.3 Graph-Matching for Semantic Parts Localization

Providing global context information and disentangling relationships is useful to distinguish fine-grained parts. For instance, upper and lower arms share highly similar appearance. To differentiate between them, global and reciprocal information, like the relationship with neighboring parts, provides an effective context prior. Hence, to further enhance the accuracy of part parsing, we tackle part-level ambiguity and localization by proposing a novel module based on an adjacency graph that matches the parts spatial relationships between ground truth and predicted parts. More in detail, the graphs capture the adjacency relationships between each couple of parts and then we enforce the matching between the ground truth and predicted graph through an additional loss term. Although graph matching is a very well studied problem [225, 226], it has never been applied to this context before, *i.e.* as a loss function to drive deep learning architectures for semantic segmentation. The only other attempt to design a graph matching loss is [227], which however deals with a completely different task, *i.e.*, domain adaptation in classification, and has a different interpretation of the graph, that measures the matching between the source and target domains.

An overview of this module is presented in Figure 6.12. Formally, we represent the graphs using two (square) weighted adjacency matrices of size N_p:

$$\tilde{\mathbf{M}}^{GT} = \left\{\tilde{m}^{GT}_{i,j}\right\}_{\substack{i=1,\dots,N_p \\ j=1,\dots,N_p}} \qquad \tilde{\mathbf{M}}^{pred} = \left\{\tilde{m}^{pred}_{i,j}\right\}_{\substack{i=1,\dots,N_p \\ j=1,\dots,N_p}} \tag{6.8}$$

The first matrix ($\tilde{\mathbf{M}}^{GT}$) contains the adjacency information computed on ground truth data, while the second ($\tilde{\mathbf{M}}^{pred}$) has the same information computed on the predicted segmentation maps. Each element of the matrices provide a measure of how close the two parts p_i and p_j are in the ground truth and in the predicted segmentation maps, respectively. We do not consider self-connections, hence $\tilde{m}^{GT}_{i,i} = \tilde{m}^{pred}_{i,i} = 0$ for $i = 1, ..., N_p$. To measure the closeness between couples of parts, that is a hint of the strength of connection between them, we consider weighted matrices where each entry $\tilde{m}_{i,j}$ depends on the length of the contour in common between them. Actually, to cope for some inaccuracies inside the dataset where some adjacent parts are separated by thin background regions, the entries of the matrices are the counts of pixels belonging to one part with a distance less or equal than T from a sample belonging to the other part. In other words, $\tilde{m}^{GT}_{i,j}$ represents the number of pixels in p_i whose distance from a pixel in p_j is less than T. We empirically set $T = 4$ pixels. Since the matrix $\tilde{\mathbf{M}}^{pred}$ needs to be recomputed at each training step, we approximate this operation by dilating the two masks of $\lceil T/2 \rceil$ and computing the intersecting region. Formally, defining with $p^{GT}_i = \mathbf{Y}_p[i]$ the mask of the i-th part in the ground truth map $\mathbf{Y}_p$, we have:

$$\tilde{m}^{GT}_{i,j} = \left| \left\{ s \in \Phi\left(p^{GT}_i\right) \cap \Phi\left(p^{GT}_j\right) \right\} \right| \tag{6.9}$$

Where s is a generic pixel, $\Phi(\cdot)$ is the morphological 2D dilation operation and $|\cdot|$ is the cardinality of the given set. We apply a row-wise L2 normalization and we obtain a matrix of *proximity ratios* $\mathbf{M}^{GT}_{[i,:]} = \tilde{\mathbf{M}}^{GT}_{[i,:]} / \|\tilde{\mathbf{M}}^{GT}_{[i,:]}\|_2$ that measures the flow from the considered part to all the others.

With this definition, non-adjacent parts have 0 as entry. The same approach is used for the adjacency matrix computed on the predicted segmentation map $\mathbf{M}^{pred}$ by substituting p^{GT}_i with $p^{pred}_i = \hat{\mathbf{Y}}_p[i]$.

Then, we simply define the Graph-Matching loss as the Frobenius norm between the two adjacency matrices:

$$\mathcal{L}_{GM} = ||\mathbf{M}^{GT} - \mathbf{M}^{pred}||_F \tag{6.10}$$

The aim of this loss function is to faithfully maintain the reciprocal relationships between parts. On one hand, disjoint parts are enforced to be predicted as disjoint; on the other hand, neighboring parts are enforced to be predicted as neighboring and to match the proximity ratios.

Summarizing, the overall training objective of our framework is:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{GM} \tag{6.11}$$

where the hyper-parameters λ_1 and λ_2 are used to control the relative contribution of the three losses to the overall objective function.

6.3.4 Training of the Deep Learning Architecture

6.3.4.1 Multi-Part Dataset

For the experimental evaluation of the proposed multi-class part parsing framework we employed the Pascal-Part [195] dataset, which is currently the largest dataset for this purpose. It contains a total of 10103 variable-sized images with pixel-level parts annotation on the 20 Pascal VOC2010 [114] semantic object classes (plus the *background* class). We employ the original split from [195] with 4998 images in the *trainset* for training and 5105 images in the *valset* for testing. We consider two different sets of labels for this dataset. Firstly, following [210], which is the only work dealing with the multi-class part parsing problem, we grouped the original semantic classes

into 58 part classes in total. Additionally, to further test our method on a even more challenging scenario, we consider the grouping rules proposed by [228] for part detection that, instead, leads to a larger set of 108 parts.

6.3.4.2 Training Details

The modules introduced in this work are agnostic to the underlying network architecture and can be extended to other scenarios. For a fair comparison with [210] we employ a DeepLab-v3 [6] architecture with ResNet101 [152] as the backbone. We follow the same training schemes of [5,6,210] and we started from the official TensorFlow [125] implementation of the DeepLab-v3 [6,229]. The ResNet101 was pre-trained on ImageNet [121] and its weights are available at [229]. During training, images are randomly left-right flipped and scaled of a factor from 0.5 to 2 times the original resolution with bilinear interpolation. The results in the testing stage are reported at the original image resolution. The model is trained for $50K$ steps with the base learning rate set to $5 \cdot 10^{-3}$ and decreased with a polynomial decay rule with power 0.9. We employ weight decay regularization of 10^{-4}. The atrous rate in the Atrous Spatial Pyramid Poooling (ASPP) is set to $(6, 12, 18)$ as in [5,210]. We use a batch size of 10 images and we set $\lambda_1 = 10^{-3}$ and $\lambda_2 = 10^{-1}$ to balance part segmentation. For the evaluation metric, we employ the mean Intersection over Union (mIoU) since pixel accuracy is dominated by large regions and little sensitive to the segmentation on many small parts, that are instead the main target of this work. The code and the part labels are publicly available at `https://lttm.dei.unipd.it/paper_data/GMNet`.

6.3.5 Experimental Results

In this section we show the experimental results on the multi-class part parsing task in two different scenarios with 58 and 108 parts respectively. We also present some ablation studies to verify the effectiveness of the proposed methodologies.

Table 6.5: IoU results on the Pascal-Part-58 benchmark. mIoU: mean per-part-class IoU. Avg: average per-object-class mIoU.

Method	bgr	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	d. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mIoU**	**Avg.**
SegNet [214]	85.4	13.7	40.7	11.3	21.7	10.7	36.7	26.3	28.5	16.6	8.9	16.6	24.2	18.8	44.7	35.4	16.1	17.3	15.7	41.3	26.1	24.4	26.5
FCN [1]	87.0	33.9	51.5	37.7	47.0	45.3	50.8	39.1	45.2	29.4	31.2	32.5	42.4	42.2	58.2	40.3	38.3	43.4	35.7	66.7	44.2	42.3	44.9
DeepLab [5]	89.8	40.7	58.1	43.8	53.9	44.5	62.1	45.1	52.3	36.6	41.9	38.7	49.5	53.9	66.1	49.0	45.3	45.3	40.5	76.8	56.5	49.9	51.9
BSANet [210]	91.6	50.0	65.7	54.8	60.2	49.2	70.1	53.5	63.8	36.5	52.8	43.7	58.3*	66.0	71.6*	58.4	55.0	49.6	43.1	82.2	61.4	58.2	58.9*
Baseline [6]	91.1	45.7	63.2	49.0	54.4	49.8	67.6	49.2	59.8	35.4	47.6	43.0	54.4	62.0	68.0	55.0	48.9	45.9	43.2	79.6	57.7	54.4	55.7
GMNet	92.7	46.7	66.4	52.0	70.0	55.7	71.1	52.2	63.2	51.4	54.8	51.3	59.6	64.4	73.9	56.2	56.2	53.6	56.1	85.0	65.6	59.0	61.8

*: values different from [210] since they were wrongly reported in the paper.

6.3.5.1 Pascal-Part-58

To evaluate our framework we start from the scenario with 58 parts, *i.e.*, the same experimental setting used in [210]. In Table 6.5 we compare the proposed model with existing semantic segmentation schemes. As evaluation criteria we employ the mean IoU of all the parts (*i.e.*, mIoU), the average IoU for all the parts belonging to each single object, and the mean of these values (denoted as Avg., *i.e.*, in this case each object has the same weight independently of the number of parts). As expected, traditional semantic segmentation architectures such as FCN [1], SegNet [214] and DeepLab [5] are not able to perform a fully satisfactory part-parsing.

We adopt as our baseline network the DeepLab-v3 architecture [6], that is the best performing among the compared standard approaches achieving 54.4% of mIoU. The proposed GMNet approach combining both the object-level semantic embedding and the graph matching module achieves a higher accuracy of 59.0% of mIoU, significantly outperforming all the other methods and in particular the baseline on every class with a gap of 4.6% of mIoU. The only other method specifically addressing part-based semantic segmentation is BSANet [210], which achieves a lower mIoU of 58.2%. Our method achieves higher results over most of the objects, both with many parts (like *cow*, *dog* and *sheep*) and with no or few parts (like *boat*, *bottle*, *chair*, *dining table* and *sofa*).

Qualitative results are shown in Figure 6.13. The effects of the two main components of our work, namely the object-level semantic embedding network $\mathcal{S}$ and the graph matching module, are clearly visible in the images.

From one side, the object-level semantic embedding network brings useful additional information prior to the part-level decoding stage, thus enriching the extracted features to be object discriminative. We can appreciate this aspect from the first and the third row. In the first row, the baseline completely misleads a dog with a cat (light green corresponds to *cat_head* and green to *cat_torso*). BSANet is able to partially recover the *dog_head* (amaranth corresponds to the proper labeling). Our method, instead, is able to accurately detect and segment the dog parts (*dog_head* in amaranth and *dog_torso* in blue) thanks to object-level priors coming from the semantic embedding module. A similar discussion can be done also on the third image, where the baseline confuses car parts (green corresponds to *window*, aquamarine to *body* and light green to *wheel*) with bus parts (pink is the *window*, brown the *body* and dark green the *wheel*) and BSANet is not able to correct this error. GMNet, instead, can identify the correct object-level class and the respective parts, excluding the very small and challenging *car_wheels*, and at the same time can better segment the *bus_window*. In row 6, BSANet confuses cow's parts with sheep's parts. In row 7, the baseline confuses sheep's parts with cat's ones and BSANet with dog's parts. GMNet is able to correctly deal with these situations thanks to the object-level guidance. Again, in the last row both the baseline and BSANet mislead the dog's parts with cat's parts, while GMNet is able to avoid this error.

From the other side, the graph matching module helps in the mutual localization of parts within the same object-level class and it is more appreciable on small parts. The effect of the graph matching module is more evident in the second and fourth row. In the second image, we can verify how both the baseline and BSANet are not able to correctly place the *dog_tail* (in yellow) misleading it with the *dog_head* (in red). Thanks to the graph matching module, GMNet can disambiguate between such parts and correctly exploit their spatial relationship with respect to the *dog_body*. In the fourth image, both the baseline and BSANet tend to overestimate the presence of the *cat_legs* (in dark green) and they miss one *cat_tail*. The constraints on the relative position among the various parts enforced by the graph matching module allow GMNet to properly segment and label the *cat_tail* and to partially correct the estimate of the *cat_legs*.

Moreover, in a very challenging image in row 5, where both the baseline and BSANet partially or completely miss some classes, our method generate superior quality segmentation maps. A vehicle behind a metal grid is being correctly identified and quite well localized in all its parts thanks to the semantic embedding module and to graph matching.

Per-part-class IoU and PA on the Pascal-Part-58 dataset are shown in Table 6.6, where it is possible to see that the proposed method (GMNet) outperforms the baseline [6] approach on almost every part both considering the per-part-IoU and the per-part-PA. With respect to BSANet [210], GMNet can produce clearly higher results on 15 objects out of 21 (such as *bottle*, *bus*, *dog*, *sheep*,...) and can produce comparable results on 2 objects (*i.e.*, on *car* ad *cat*).

We can further verify the ranking of the compared methods analyzing the **average metrics**

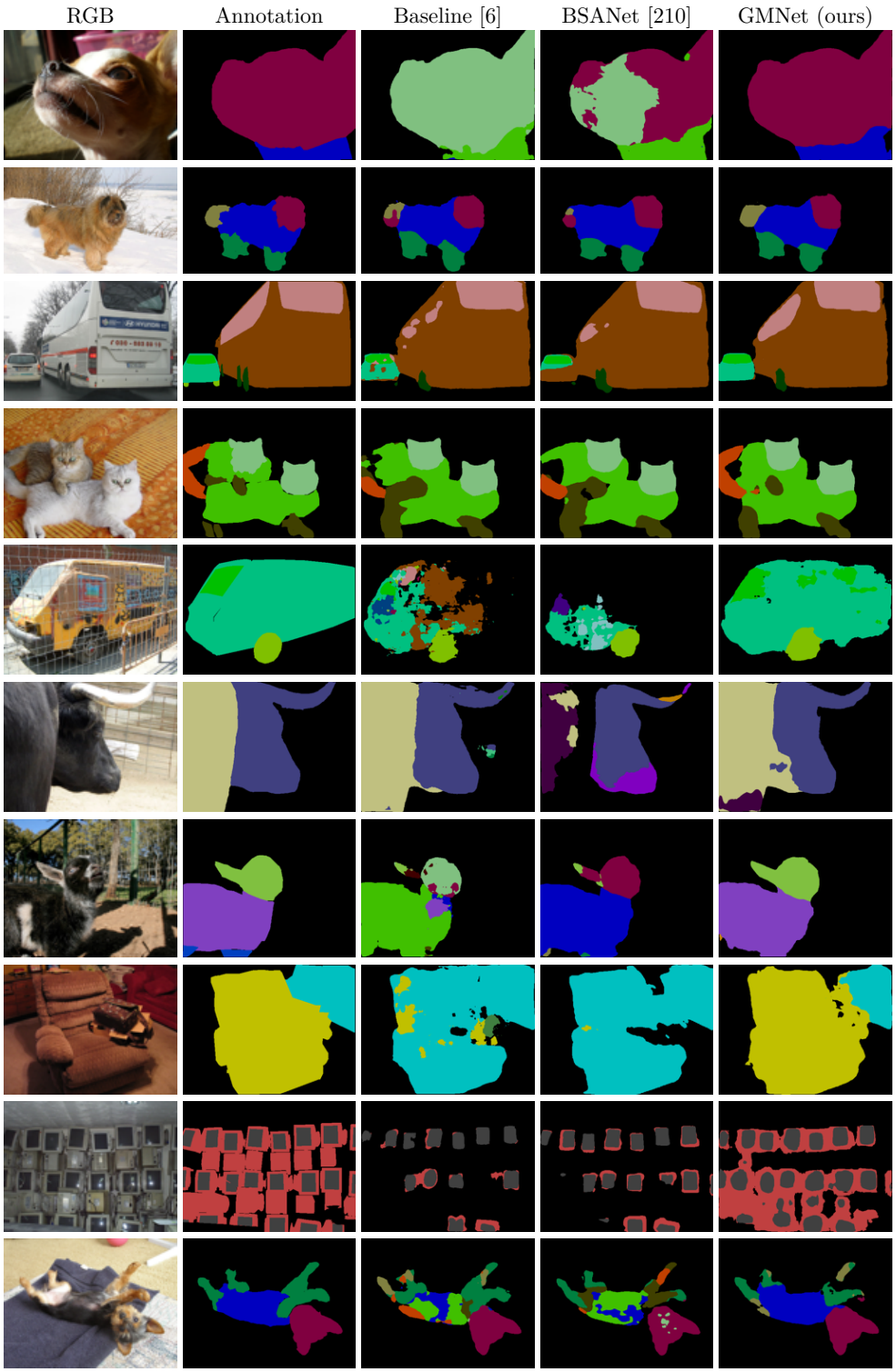


Figure 6.13: Qualitative results on some sample scenes on the Pascal-Part-58 dataset.

reported in Table 6.7. Here, we can appreciate how GMNet is able to outperform both the baseline and BSANet robustly on all the most widely used metrics for semantic segmentation.

Table 6.6: Per-part IoU and PA on the Pascal-Part-58 dataset.

Parts Name	Baseline		BSANet		GMNet	
	IoU	PA	IoU	PA	IoU	PA
background	91.1	96.3	91.6	96.7	**92.7**	**96.9**
aeroplane body	66.6	79.8	**70.0**	**81.4**	69.6	81.2
aeroplane engine	25.7	31.4	**29.1**	**33.8**	25.7	31.2
aeroplane wing	33.5	48.2	**38.3**	**49.1**	34.2	46.4
aeroplane stern	57.1	68.2	**59.2**	**72.5**	57.2	70.8
aeroplane wheel	45.4	53.3	**53.2**	**62.5**	46.8	53.3
bike wheel	78.0	88.1	78.0	**88.6**	**81.3**	88.5
bike body	48.4	61.2	**53.4**	**68.4**	51.5	64.2
bird head	64.6	72.7	**74.0**	**80.2**	71.1	79.3
bird wing	35.1	45.5	**39.7**	**53.2**	38.6	52.9
bird leg	29.3	37.6	**34.8**	**42.6**	28.7	35.4
bird torso	66.9	83.1	**70.9**	**84.4**	69.5	83.1
boat	54.4	64.8	60.2	69.6	**70.0**	**78.5**
bottle cap	30.7	35.4	29.8	35.0	**33.9**	**42.5**
bottle body	68.8	78.5	68.6	74.8	**77.6**	**86.1**
bus window	72.7	83.7	74.8	85.9	**75.4**	**86.1**
bus wheel	55.3	66.3	57.1	70.1	**58.1**	**72.1**
bus body	74.8	88.2	78.3	88.7	**79.9**	**89.8**
car window	62.6	73.9	**68.1**	**78.2**	64.8	77.5
car wheel	64.8	78.1	68.5	79.7	**70.3**	**79.8**
car light	46.2	54.3	**53.7**	**61.7**	48.4	56.0
car plate	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
car body	72.1	86.4	77.0	**88.4**	**77.6**	88.2
cat head	80.2	90.4	83.7	**92.3**	**83.8**	91.6
cat leg	48.6	**61.2**	**50.1**	58.6	49.4	59.1
cat tail	40.2	51.3	**48.8**	55.6	46.0	**56.7**
cat torso	70.3	85.7	72.6	**88.0**	**73.8**	87.6
chair	35.4	43.3	36.5	42.7	**51.4**	**63.9**
cow head	74.3	85.6	76.4	86.0	**80.7**	**87.8**
cow tail	0.0	0.0	7.9	8.1	**8.1**	**8.4**
cow leg	46.1	62.3	53.4	**67.5**	**53.5**	67.2
cow torso	69.9	83.5	73.5	85.9	**77.1**	**87.8**
dining table	43.0	55.4	43.7	54.8	**51.3**	**62.6**
dog head	78.7	88.3	82.5	91.4	**85.0**	**92.7**
dog leg	48.1	59.9	53.8	63.0	**53.8**	**64.8**
dog tail	27.1	39.4	31.3	38.0	**31.4**	**41.5**
dog torso	63.7	76.8	65.7	79.7	**68.0**	**81.2**
horse head	74.7	81.7	**76.6**	**83.3**	73.9	80.5
horse tail	47.0	60.4	**51.0**	59.9	50.4	**62.2**
horse leg	55.9	70.9	**61.6**	**75.8**	59.3	72.9
horse torso	70.3	84.2	**74.9**	86.6	73.9	**87.4**
mbike wheel	70.9	82.5	71.6	82.1	**73.5**	**84.0**
mbike body	65.1	80.9	71.5	87.7	**74.3**	**87.8**
person head	83.5	91.6	**85.0**	**92.3**	84.7	91.8
person torso	65.9	80.6	**68.2**	**82.7**	67.0	82.3
person larm	46.9	60.0	**52.0**	**65.6**	48.6	62.8
person uarm	51.5	65.8	**54.4**	**68.2**	52.4	66.9
person lleg	38.6	51.5	**43.5**	**54.6**	40.2	51.5
person uleg	43.8	60.0	**47.4**	**63.5**	44.5	59.9
pplant pot	45.3	61.0	53.5	64.8	**56.0**	**69.1**
pplant plant	52.4	62.1	**56.6**	65.8	56.4	**66.4**
sheep head	60.9	69.3	65.4	71.3	**70.8**	**79.0**
sheep leg	8.6	11.1	11.7	16.5	**14.3**	**20.2**
sheep torso	68.3	84.4	71.6	86.1	**75.6**	**88.7**
sofa	43.2	58.8	43.1	57.4	**56.1**	**65.0**
train	79.6	86.1	82.2	90.2	**85.0**	**92.0**
tv screen	69.5	76.0	73.1	78.6	**77.0**	**84.3**
tv frame	45.9	56.9	49.8	60.9	**54.1**	**67.4**

6.3.5.2 Pascal-Part-108

To further verify the robustness and the scalability of the proposed methodology we perform a second set of experiments using an even larger number of parts. The results on the Pascal-Part-108 benchmark are reported in Table 6.8. Even though we can immediately verify a drop in the overall performance, that is predictable being the task more complex with respect to the previous scenario with an almost double number of parts, we can appreciate that our framework is able to largely surpass both the baseline and [210]. It achieves a mIoU of 45.8%, outperforming the baseline by 4.5% and the other compared standard segmentation networks by an even larger margin. The gain with respect to the main competitor [210] is remarkable with a gap of 2.9%

Table 6.7: Comparison in terms of mIoU, mCA and mPA on Pascal-Part-58.

Method	mIoU	mPA	mCA
Baseline [6]	54.45	89.86	65.42
BSANet [210]	58.15	90.76	68.12
GMNet	**59.04**	**91.55**	**69.22**

of mIoU. In this scenario, indeed, most of the previous considerations holds and are even more evident from the results. The gain in accuracy is stable across the various classes and parts: the proposed framework significantly wins by large margins on almost every per-object-class mIoU.

Thanks to the object-level semantic embedding network our model is able to obtain accurate segmentation of all the objects with few or no parts inside, such as *boat, bottle, chair, plant* and *sofa*. On these classes, the gain with respect to [210] ranges from 5.4% for the *plant* class to an impressive 15% on the *chair* class. On the other hand, thanks to the graph matching module, our framework is also able to correctly understand the spatial relationships between small parts, as for example the ones contained in *cat, cow, horse* and *sheep*. Although objects are composed by tiny and difficult parts, the gain with respect to [210] is still significant and ranges between 1.5% on *horse* parts to 11.2% on *cow* ones.

The qualitative results for some sample scenes presented in Figure 6.14 confirm the numerical evaluation. We can appreciate that the proposed method is able to compute accurate segmentation maps both when a few elements or many parts coexist in the scene. More in detail, in the first row we can verify the effectiveness of the object-level semantic embedding in conditioning part parsing. The baseline is not able to localize and segment the body and the neck of the sheep. The BSANet approach [210] achieves even worse segmentation and labeling performance. Such methods mislead the sheep with a dog (in the figure light blue denotes *dog_head*, light purple *dog_neck*, brown *dog_muzzle* and yellow *dog_torso*) or with a cat (purple denotes *cat_torso*). Thanks to the object-level priors, GMNet is able to associate the correct label to each of the parts correctly identifying the sheep as the macro class. In the second row, the effect of the graph matching procedure is more evident, which is much more significant on this dataset because it contains many small-sized parts. The baseline approach tends to overestimate and badly localize the *cow_horns* (in brown) and BSANet confuses the *cow_horns* with the *cow_ears* (in pink). GMNet, instead, achieves superior results thanks to the graph module which accounts for proper localization and contour shaping of the various parts. In the third row, a scenario with two object-level classes having no sub-parts is reported. Again, we can check how GMNet is able to discriminate between *chair* (in pink) and *sofa* (in light brown). In the fourth row we can appreciate how the two parts of the *potted plant* are correctly segmented by GMNet thanks to the semantic embedding module for what concerns object identification and to the graph

Table 6.8: IoU results on the Pascal-Part-108 benchmark. mIoU: mean per-part-class IoU. Avg: average per-object-class mIoU. †: re-trained on the Pascal-Part-108 dataset.

Method	bgr	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	d. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mIoU**	**Avg.**
SegNet [214]	85.3	11.2	32.4	6.3	21.4	10.3	27.9	22.6	22.8	17.0	6.3	12.5	21.1	14.9	12.2	32.2	13.8	12.6	15.2	11.3	27.5	18.6	20.8
FCN [1]	86.8	30.3	35.6	23.6	47.5	44.5	21.3	34.5	35.8	26.6	20.3	24.4	37.7	29.8	14.2	35.6	34.4	28.9	34.0	18.1	45.6	31.6	33.8
DeepLab [5]	90.2	38.3	35.4	29.4	57.0	41.5	27.0	40.1	45.5	36.6	33.3	35.2	41.1	48.8	19.5	40.6	46.0	23.7	40.8	17.5	70.0	35.7	40.8
BSANet† [210]	91.6	45.3	40.9	41.0	61.4	48.9	32.2	43.3	50.7	34.1	39.4	45.9	52.1	50.0	23.1	52.4	50.6	37.8	44.5	20.7	66.3	42.9	46.3
Baseline [6]	90.9	41.9	44.5	35.3	53.7	47.0	34.1	42.3	49.2	35.4	39.8	33.0	48.2	48.8	23.2	50.4	43.6	35.4	39.2	20.7	60.8	41.3	43.7
GMNet	92.7	48.0	46.2	39.3	69.2	56.0	37.0	45.3	52.6	49.1	50.6	50.6	52.0	51.5	24.8	52.6	56.0	40.1	53.9	21.6	70.7	45.8	50.5

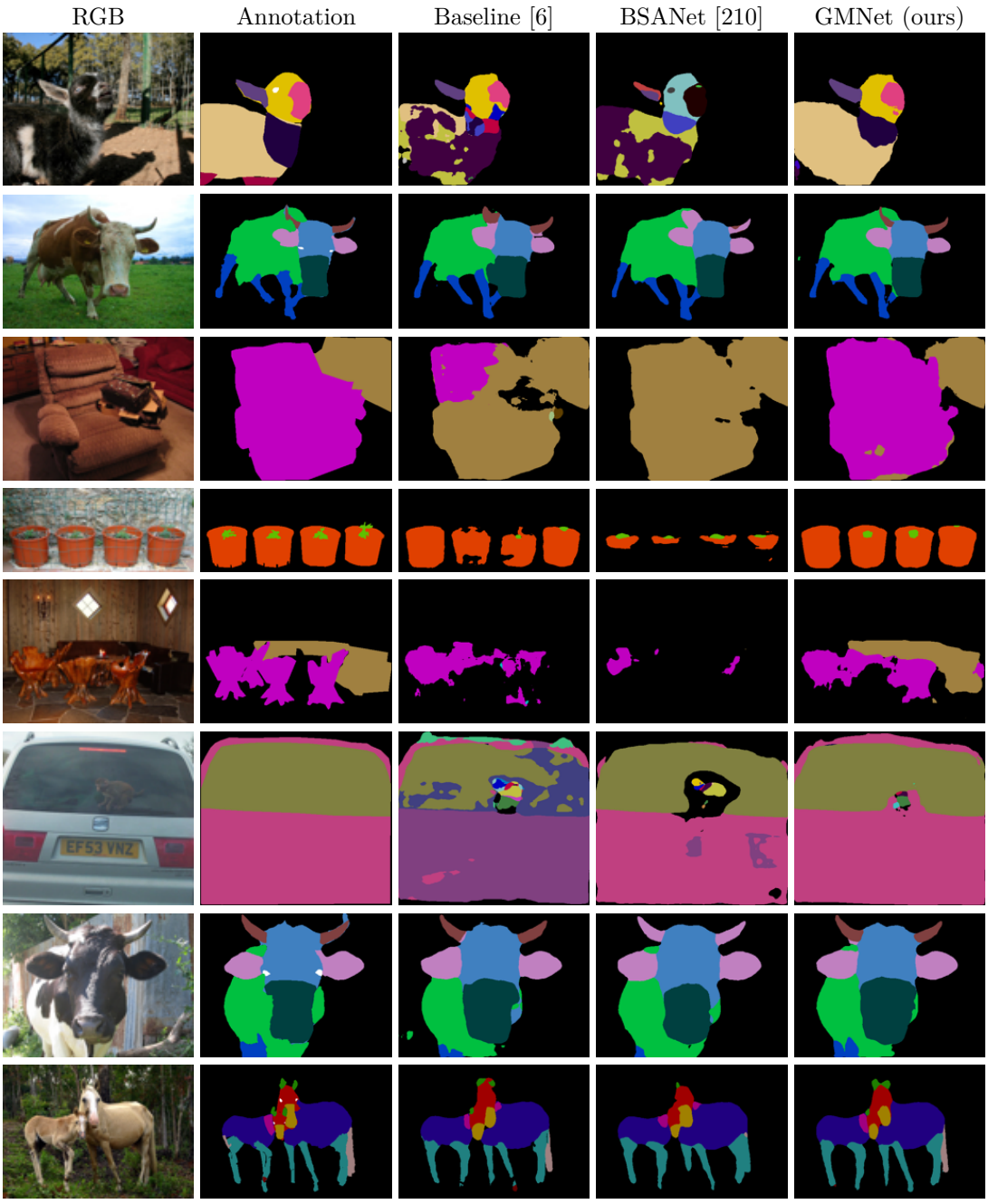


Figure 6.14: Qualitative results on sample scenes on the Pascal-Part-108 dataset.

matching strategy for what concerns small parts localization. In rows 5 and 6, GMNet generates cleaner segmentation maps exploiting object-level priors which help to disambiguate between cars and buses. In row 7, the *cow_horns* and *cow_body* are badly localized and labelled both by the baseline and by BSANet. However, the graph matching component on the reciprocal spatial relationship between these parts and the others guides the network to properly localize and label such parts. In row 8 our framework is able to well localize horse's parts and especially the challenging *horse_tail* part.

Per-part-IoU and per-part-PA on the Pascal-Part-108 dataset are reported in Table 6.9, where we can notice that the gap between the proposed framework and the compared methods is significantly larger than for the Pascal-Part-58 dataset. GMNet achieves higher accuracy than the competitors on almost all the parts. In particular, our framework is able to outperform BSANet [210] in 19 out of 21 object-level classes both with many parts within them (such as *aeroplane*, *bus*, *cat*, *dog*, *person*, *sheep*,...) and with no or few parts within them (such as *boat*, *bottle*, *chair*, *sofa*, *tv*,..).

The mean accuracy results are shown in Table 6.10 where we can verify that our method clearly outperforms both the baseline [6] and BSANet [210] on all the most popular metrics used to evaluate semantic segmentation architectures. Hence, we prove the robustness of our framework to different evaluation criteria and to different datasets. Additionally, we argue that the proposed framework is able to scale well to even larger sets of parts.

6.3.6 Ablation Studies

In this section we conduct an accurate investigation of the effectiveness of the various modules of the proposed work on the Pascal-Part-58 dataset.

We start by evaluating the individual impact of the modules and the performance analysis is shown in Table 6.11. Let us recall that the baseline architecture (*i.e.*, the DeepLab-v3 network trained directly on the 58 parts with only the standard cross-entropy loss enabled) achieves a mIoU of 54.4%. The reconstruction loss on the object-level segmentation maps helps in preserving the object-level shapes rearranging parts into object-level classes and allows to improve the mIoU to 55.2%. The semantic embedding network $\mathcal{S}$ acts as a powerful class-conditioning module to retain object-level semantics when learning parts and allows to obtain a large performance gain: its combination with the reconstruction loss leads to a mIoU of 58.4%. The addition of the graph matching procedure further boost the final accuracy to 59.0% of mIoU. To better understand the contribution of this module we also tried a simpler unweighted graph model whose entries are just binary values representing whether two parts are adjacent or not (column $\mathcal{L}^u_{GM}$ in the table). This simplified graph leads to a mIoU of 58.7%, lacking some information about the closeness of adjacent parts.

Then, we present a more accurate analysis of the impact of the semantic embedding module and the results are summarized in Table 6.12. First of all, the exploitation of the multiple concatenation between features computed by $\mathcal{S}$ and features of $\mathcal{D}_p$ at different resolutions allows object-level embedding at different scales and enhances the scale invariance. Concatenating only the output of $\mathcal{S}$ with the output of $\mathcal{E}_p$ (we refer to this approach with "single concatenation"), the final mIoU slightly decreases to 58.7%. In order to evaluate the usefulness of exploiting features extracted from a CNN, we compared the proposed framework with a variation directly concatenating the output of $\mathcal{E}_p$ with the object-level predicted segmentation maps $\hat{\mathbf{Y}}_o$ after a proper rescaling ("without $\mathcal{S}$"). This approach leads to a quite low mIoU of 55.7%, thus outlining that the embedding network $\mathcal{S}$ is very effective and that a simple stacking of architectures is not the best option for our task. Additionally, we considered also the option of directly feeding object-level features to the part parsing decoder, *i.e.*, we tried to concatenate the output of $\mathcal{E}_o$

Table 6.9: Per-part IoU and PA on the Pascal-Part-108 dataset.

Parts Name	Baseline		BSANet		GMNet		Parts Name	Baseline		BSANet		GMNet	
	IoU	PA	IoU	PA	IoU	PA		IoU	PA	IoU	PA	IoU	PA
background	90.9	**97.2**	91.6	97.1	**92.7**	97.0	dining table	33.0	40.2	45.9	59.7	**50.6**	**62.3**
aero body	61.9	72.3	**68.2**	77.6	61.9	**82.6**	dog head	60.5	75.5	63.8	78.2	**64.0**	**78.9**
aero stern	53.2	68.4	54.2	65.3	**57.4**	**71.0**	dog reye	50.1	61.4	54.1	61.4	**54.7**	**64.7**
aero rwing	28.9	39.8	33.1	**46.5**	**34.3**	46.0	dog rear	54.0	69.4	**57.2**	73.4	56.8	**73.9**
aero engine	24.7	29.0	26.5	32.0	**27.2**	**32.6**	dog nose	63.5	75.0	**66.3**	74.3	66.0	**76.8**
aero wheel	40.9	46.8	44.5	49.6	**51.5**	**61.3**	dog torso	58.4	74.6	62.3	78.4	**63.2**	**79.1**
bike fwheel	78.4	85.7	75.3	86.7	**80.2**	**87.8**	dog neck	27.1	35.4	26.2	30.8	**28.1**	**35.5**
bike saddle	34.1	39.8	31.0	31.9	**38.0**	**43.2**	dog rfleg	39.2	50.6	42.4	53.5	**43.7**	**55.8**
bike handlebar	**23.3**	**26.1**	20.6	22.8	22.4	25.9	dog rfpaw	39.4	47.9	**44.2**	51.7	43.7	**52.9**
bike chainwheel	42.3	50.4	36.5	41.6	**44.1**	**57.0**	dog tail	24.7	37.8	**34.9**	**42.3**	30.8	41.4
birds head	51.5	61.3	**66.4**	**78.0**	65.3	77.7	dog muzzle	65.1	76.1	**69.4**	**82.3**	68.9	80.4
birds beak	40.4	49.5	**47.1**	**54.6**	44.3	54.0	horse head	54.4	67.0	**57.1**	**68.9**	55.9	68.3
birds torso	61.7	77.9	**65.2**	79.4	64.8	**82.6**	horse rear	49.7	58.1	51.1	56.5	**52.2**	**65.6**
birds neck	27.5	32.2	**39.1**	**50.1**	28.4	35.7	horse muzzle	61.3	68.7	**65.2**	**74.0**	62.9	69.5
birds rwing	35.9	50.4	**39.3**	**53.7**	37.2	50.1	horse torso	56.7	75.9	59.5	75.9	**60.7**	**84.3**
birds rleg	23.5	28.6	**26.5**	32.2	23.8	**32.8**	horse neck	42.1	51.3	**49.6**	**64.8**	47.2	55.8
birds rfoot	13.9	16.3	11.6	12.7	**17.7**	**22.5**	horse rfuleg	54.1	68.5	**57.0**	**71.8**	56.4	70.9
birds tail	28.1	39.2	**33.0**	44.1	32.5	**46.1**	horse tail	48.1	63.5	47.6	54.5	**51.4**	**64.4**
boat	53.7	60.3	61.4	71.5	**69.2**	**77.8**	horse rfho	24.1	31.4	12.9	13.7	**25.3**	**32.7**
bottle cap	30.4	35.0	26.2	30.0	**33.4**	**40.0**	mbike fwheel	69.6	78.9	69.3	80.4	**73.6**	**83.3**
bottle body	63.7	69.5	71.5	78.3	**78.7**	**88.3**	mbike hbar	**0.0**	**0.0**	0.0	0.0	0.0	0.0
bus rightside	70.8	85.3	73.0	83.7	**75.7**	**88.4**	mbike saddle	0.0	0.0	0.0	0.0	**0.8**	**0.8**
bus roofside	7.5	7.7	0.3	0.3	**13.5**	**14.4**	mbike hlight	25.8	**32.8**	10.6	11.2	**28.5**	32.4
bus mirror	2.1	2.2	0.3	0.3	**6.6**	**7.6**	person head	68.2	81.9	**69.7**	82.2	69.3	**82.7**
bus fliplate	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	person reye	35.1	39.3	**41.3**	**46.3**	38.7	43.9
bus door	**40.1**	51.2	37.2	**53.2**	38.1	47.3	person rear	37.4	46.0	**41.9**	49.4	41.4	**51.5**
bus wheel	**54.8**	65.5	53.1	63.9	56.7	**69.4**	person nose	53.0	62.1	54.3	63.1	**56.7**	**67.5**
bus headlight	**25.6**	28.3	19.9	20.8	30.4	**34.2**	person mouth	48.9	56.9	49.5	54.9	**51.3**	**60.8**
bus window	71.8	85.2	73.5	86.4	**74.6**	**87.4**	person hair	70.8	83.3	**72.3**	**85.9**	71.8	83.9
car rightside	64.0	78.0	67.9	81.2	**70.5**	**84.5**	person torso	63.4	79.1	64.3	78.3	**65.2**	**80.9**
car roofside	21.0	25.4	16.1	17.6	**22.3**	**26.6**	person neck	49.7	63.8	50.9	65.1	**51.2**	**65.3**
car fliplate	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	person ruarm	54.7	68.6	55.7	70.2	**57.4**	**71.3**
car door	41.4	52.5	39.6	49.0	**42.3**	**53.5**	person rhand	43.0	55.4	**47.4**	**57.6**	44.1	56.8
car wheel	65.8	74.5	64.0	76.6	**70.2**	**80.0**	person ruleg	50.8	66.0	52.3	67.1	**53.0**	**67.9**
car headlight	42.9	48.4	**49.4**	**59.7**	46.4	54.4	person rfoot	29.8	38.9	28.9	32.4	**31.3**	**39.8**
car window	61.0	75.5	**66.5**	**82.4**	65.0	79.0	pplant pot	43.6	54.5	50.6	58.9	**56.0**	**69.0**
cat head	73.9	87.3	75.6	88.5	**77.5**	**88.5**	pplant plant	42.9	48.8	55.5	**68.7**	**56.6**	66.6
cat reye	58.8	69.0	62.0	71.1	**62.8**	**71.8**	sheep head	45.6	56.9	47.0	58.0	**54.0**	**66.9**
cat rear	65.5	77.7	66.8	77.1	**67.1**	**78.8**	sheep rear	43.2	53.0	**47.7**	56.6	45.3	**58.2**
cat nose	40.3	49.1	41.2	45.8	**46.3**	**56.2**	sheep muzzle	58.2	67.0	61.1	72.4	**64.9**	**74.7**
cat torso	64.2	81.4	66.8	84.2	**68.7**	**86.0**	sheep rhorn	3.0	3.6	0.0	0.0	**5.4**	**6.6**
cat neck	22.8	33.8	19.8	25.0	24.4	**34.1**	sheep torso	62.6	78.0	66.4	83.6	**68.8**	**86.3**
cat rfleg	36.5	48.5	38.5	49.2	**39.1**	**50.0**	sheep neck	26.9	38.1	25.3	**41.2**	**30.3**	41.0
cat rfpaw	40.6	50.2	**43.4**	**51.5**	41.7	50.7	sheep rfuleg	8.6	10.6	**17.4**	**24.5**	11.7	14.7
cat tail	40.2	52.2	42.6	49.5	**45.8**	**57.0**	sheep tail	6.7	7.4	1.1	1.1	**9.1**	**11.5**
chair	35.4	42.3	34.1	38.4	**49.1**	**60.4**	sofa	39.2	50.7	44.5	56.9	**53.9**	**66.1**
cow head	51.2	65.5	58.2	74.2	**63.8**	**74.9**	train head	5.3	**6.4**	**5.6**	6.4	4.5	5.3
cow rear	51.2	68.5	53.0	72.9	**60.0**	**75.1**	train hrightside	61.9	77.3	**63.5**	**84.0**	60.8	83.1
cow muzzle	61.2	77.6	67.2	81.9	**74.9**	**86.7**	train hroofside	**23.0**	**28.0**	13.7	17.0	21.1	26.3
cow rhorn	28.8	35.0	10.1	10.2	**44.0**	**50.6**	train headlight	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
cow torso	63.4	78.6	69.9	85.8	**73.2**	**87.2**	train coach	28.6	33.6	**42.0**	**47.8**	31.4	37.9
cow neck	9.5	12.7	7.3	7.9	**20.3**	**25.9**	train crightside	15.6	24.5	**19.0**	30.6	14.9	**33.8**
cow rfuleg	46.5	60.0	49.7	61.4	**54.8**	**70.7**	train croofside	10.8	11.9	1.0	1.0	**18.1**	**22.6**
cow tail	6.5	7.3	0.1	0.1	**13.6**	**14.9**	tv screen	60.8	71.3	66.3	79.5	**70.7**	**82.9**

Table 6.10: Comparison in terms of mIoU, mCA and mPA on Pascal-Part-108.

Method	mIoU	mPA	mCA
Baseline [6]	41.36	88.57	50.51
BSANet [210]	42.95	89.52	51.71
GMNet	**45.80**	**90.32**	**55.68**

Table 6.11: mIoU ablation results on Pascal-Part-58. $\mathcal{L}^u_{GM}$: graph matching with unweighted graph.

$\mathcal{L}_{CE}$	$\mathcal{L}_{rec}$	$\mathcal{S}$	$\mathcal{L}^u_{GM}$	$\mathcal{L}_{GM}$	mIoU
✓					54.4
✓	✓				55.2
✓	✓	✓			58.4
✓	✓	✓	✓		58.7
✓	✓	✓		✓	**59.0**

Table 6.12: mIoU on Pascal-Part-58 with different configurations for the object-level semantic embedding.

Method	mIoU
Single concatenation	58.7
Without $\mathcal{S}$	55.7
$\mathcal{E}_o$ conditioning	55.7
GMNet	**59.0**
With objects GT	65.6

with the output of $\mathcal{E}_p$ and feed these features to $\mathcal{D}_p$ ("$\mathcal{E}_o$ conditioning"). Conditioning the part parsing with this approach does not bring in sufficient object-level indication and it leads to a mIoU of 55.7%, which is significantly lower than the complete proposed framework (59.0%). Finally, to estimate an upper limit of the performance gain coming from the semantic embedding module we fed the object-level semantic embedding network $\mathcal{S}$ with object-level ground truth annotations $\mathbf{Y}_o$ ("with objects GT"), instead of the predictions $\hat{\mathbf{Y}}_o$ (notice that the network $\mathcal{A}_o$ has good performance but introduces some errors, as it has 71.5% of mIoU at object-level). In this case, a mIoU of 65.6% is achieved, showing that there is still room for improvement.

We conclude remarking that GMNet achieves almost always higher accuracy than the starting baseline, even if small and unstructured parts remain the most challenging to be detected. Furthermore, the gain depends also on the amount of spatial relationships that can be exploited.

6.4 Conclusions and Future Work

In this chapter, we introduced and addressed two related problems of coarse-to-fine learning of semantic concepts.

In the first problem of hierarchical (semantic-level) coarse-to-fine learning, a deep neural network is trained on a small set of macro-classes and is then adapted and refined to recognize a larger set of classes with a finer semantic content. We proposed three different hierarchical strategies exploiting the softmax and argmax of the coarse network output and the edges information from the segmentation maps of the coarse network. Furthermore, a scheme for the joint training on the three tasks is also proposed. Experimental results show that all the proposed schemes allow improving the performance with respect to the direct training on the larger set of classes.

In the second problem, we tackled the emerging task of multi-class semantic part segmentation. We propose a novel coarse-to-fine strategy (at the spatial level) where the features extracted from a semantic segmentation network are enriched with object-level semantics when learning part-level segmentation. Additionally, we designed a novel adjacency graph-based module that aims at matching the relative spatial relationships between ground truth and predicted parts which has shown large improvements particularly on small parts. Combining the proposed methodologies we were able to achieve state-of-the-art results in the challenging task of multi-object part parsing

both at a moderate scale and at a larger one.

Future research will investigate the extension of the proposed modules to other scenarios and datasets. More advanced curriculum learning techniques will be embedded in our frameworks to boost the final accuracy values. Edge-related information coming from part-level and object-level segmentation maps could further aid the identification of the finer-grained semantic categories. Finally, novel graph representations better capturing part relationships and different matching functions will be investigated.

Part III

Semantic Recognition across New Visual Domains

7 Unsupervised Domain Adaptation (UDA)

7.1 Introduction

Over the past few years, deep learning techniques have shown impressive results and have achieved great success in many visual applications.

The standard supervised learning setting assumes the availability of a large training set containing data with the same statistical properties as the target data labeled according to the problem at hand. This learning setup has been used to design a huge number of machine learning strategies, from simple linear classifiers to advanced deep learning methods and has allowed to obtain a robust theoretical framework. However, deep learning methods typically require a huge amount of labeled data matching the considered scenario to obtain reliable performance. The collection and annotation of large datasets for every new task and domain is extremely expensive, time-consuming and error-prone. Furthermore, in many scenarios sufficient training data may not be available for various reasons, but it often happens that a large amount of data is available for other domains and tasks that are in some way related to the considered one. Hence, the ability to use a model trained on samples from a different, but correlated, task would highly benefit real-world applications for which there is scarce data [230]. These considerations are especially true for semantic segmentation, where the learning architectures require a huge amount of manually labeled data, which is extremely expensive to obtain since a per-pixel labeling is needed. This problem is more demanding but also particularly interesting since the labeling operation is highly time-consuming (much more than in image classification), thus making the construction of large training sets more difficult. Even if a huge number of strategies can be exploited for this task, nowadays most methods exploit deep learning strategies and, in particular, Convolutional Neural Networks (CNNs) with an auto-encoder structure.

In this chapter, we introduce the Unsupervised Domain Adaptation (UDA) task applied to semantic segmentation, and we cluster the most relevant frameworks from the literature according to the methodologies employed.

7.1.1 Domain Adaptation (DA)

Most machine learning models, including Neural Networks (NNs), typically assume that training and test samples are drawn according to the same distribution. However, there are many cases in practical settings where the training and the test data distributions differ. In the following we focus on the case where a model is trained in one or more domains (called source domains)

and then applied in another different, but related, domain (called target domain) [231]. Such learning task is known as Domain Adaptation (DA) and is a fundamental problem in machine learning. Nowadays, it has gained wide attention from the scientific community and represents a long-standing problem in many real-world applications, such as computer vision [232], natural language processing [233], sentiment analysis [234], email filtering, and several others.

Domain Adaptation can be regarded as a particular case of Transfer Learning (TL) that utilizes labeled data in one or more relevant source domains to execute new tasks in a target domain. The aim of DA methodologies is to address the distribution change or the domain shift, which typically greatly degrades the performance of the models [235]. Over the past decades, various DA methods have been proposed to address the shifts between the source and target domains for both traditional machine learning strategies and recent deep learning architectures. The intrinsic nature of source and target domains highly influence the final performance of the DA algorithms. Indeed, they are assumed to be somehow related to each other, but not identical. The more correlated they are, the easier the DA task becomes, allowing to achieve high results on the test data. Hence, a key ingredient for a well-performing strategy is the ability to discover suitable source data to extract useful clues from.

Good reviews of the domain adaptation field can be found in [230–232,235,236], which provide a comprehensive sight of the domain adaptation problem in its theoretical form [231, 235] or its generic application to visual tasks [230,232,236]. Our work differs from those in that it specifically addresses unsupervised domain adaptation for semantic segmentation. Indeed, we are motivated by the fact that this research area has recently attracted huge interest and a remarkable effort has been undertaken for its solution.

7.1.2 Unsupervised Domain Adaptation (UDA)

Before the deep learning revolution, semantic segmentation was a very challenging task and even sophisticated algorithms were achieving only limited performance; nowadays, with the advent of deep neural networks we can obtain remarkable results, provided that enough computational resources are available. Nonetheless, the potential stored in deep learning models can be fully unleashed only when sufficiently plentiful and carefully labeled training data is available. The complexity enclosed in the millions of learnable parameters of state-of-the-art deep learning models easily leads to overfitting of the training data, rather than to an enhanced model performance, and this has to be counteracted by using huge datasets for training. A major example of the central role played by the availability of large amounts of training data is the ImageNet large-scale dataset [121], whose contribution in the early development and expansion of deep neural networks for image classification is certainly very relevant.

Unfortunately, the collection and annotation of data samples is often extremely expensive, time consuming and error-prone, since it requires a large amount of human supervision in the process. The excessive cost may prevent from gathering enough data to address a new task or to move in a new environment, thus posing a serious threat to the remarkable advance brought by deep learning approaches. Therefore, it may be extremely beneficial to rely on previously built datasets, whenever they share similar properties to the target data. In this way, already available samples can be efficiently exploited to address the current task, since they belong to a domain correlated to the target one.

Even though the transferring of information from related domains appears quite appealing and fairly straightforward, in practice the process requires careful handling. Deep neural networks typically lack generalization skills; in other words, even a small change in data distribution between training and test statistical distributions might cause a severe drop of performance. For this reason, the simple application of a pre-trained model in a new environment is likely

to fail, as domain-specific attributes are usually captured alongside domain-invariant ones, thus preventing an effective knowledge transfer. In this scenario, Domain Adaptation comes in handy, as it allows to handle the statistical gap between source and target representations. The ultimate goal of the adaptation effort is to learn a prediction model on a selected task working optimally on both source and target domains, while supervision largely (or solely) comes from the label-abundant source domain. To this end, an efficient transfer of knowledge learned in the source domain to the target one is crucial to eventually reach an overall good performance. Particularly interesting is the Unsupervised Domain Adaptation (UDA) setting, in which target annotations are totally missing. This is an extremely favorable, yet challenging, scenario, as data from the target domain no longer requires expensive labeling.

Recently, the domain adaptation task has been very actively studied in the context of deep learning applied to visual tasks. While deep convolutional frameworks have proven to be capable of learning visual features useful to solve multiple related problems (*e.g.* image classification, object detection, semantic segmentation), the transferability of those representations typically shows to decrease when moving to deeper network layers [237].

Early works on domain adaptation for deep networks mainly focused on the image classification task. In many approaches a layer-wise measure of statistical domain discrepancy is jointly estimated and minimized, thus promoting the extraction of domain-invariant feature representations, while discrimination ability is guaranteed by a task-specific loss. Later, adversarial domain adaptation strategies have proven to be extremely successful, in which schemes the domain discrepancy is expressed in the form of a learnable discriminator and its minimization is performed in an adversarial manner. This has effectively opened the door to domain adaptation solutions apt to solve the semantic segmentation task, where the inherent higher complexity in terms of network representations needed for pixel-wise classification calls for more advanced solutions.

The domain adaptation task can be performed using only data from the source domain or using also some samples from the target domain. The simplest solution that could be adopted is to train only on labeled samples from the source domain without using data from the target domain, hoping that no adaptation is needed (source only). In practice, this leads to poor performance, even when only a small visual domain shift exists. To cope with this, UDA approaches exploit labeled samples from the source domain and unlabeled samples from the target one (source to target UDA).

In this scenario, the typical assumption is that the source and target domains are different but in some way related (*e.g.*, the source could be synthetically generated data resembling the real-world representations in the target one). Typically, an initial supervised training on the source domain is adapted to the target one by means of various unsupervised learning strategies aiming at achieving good performance also on the target domain (for which no labels are available). In the standard setting, the set of target classes are the same, but advanced settings where also the target labels change can be considered (see Section 7.2.1). Figure 7.1 shows the typical flow of source and target data in the unsupervised domain adaptation process.

As we observed, in domain adaptation the focus lies on the input domain distributions where, typically, a single domain shift is performed all at a sudden. In continual learning, instead, the joint domain distribution changes multiple times over time, thus representing different tasks (see Chapter 2). Such changes are typically represented by an expansion of the label set, by adding new labels or by splitting the existing ones into more refined sub-classes. One of the main targets is the capability to adapt the network to the new setting using only data concerning the new tasks and without re-training the model from scratch. However, this is highly nontrivial due to the so-called *catastrophic forgetting* phenomenon, *i.e.*, a machine learning model tends to forget knowledge about previous tasks when it learns new ones.

Hybrid approaches combining Unsupervised Domain Adaptation (UDA) and Continual Learn-

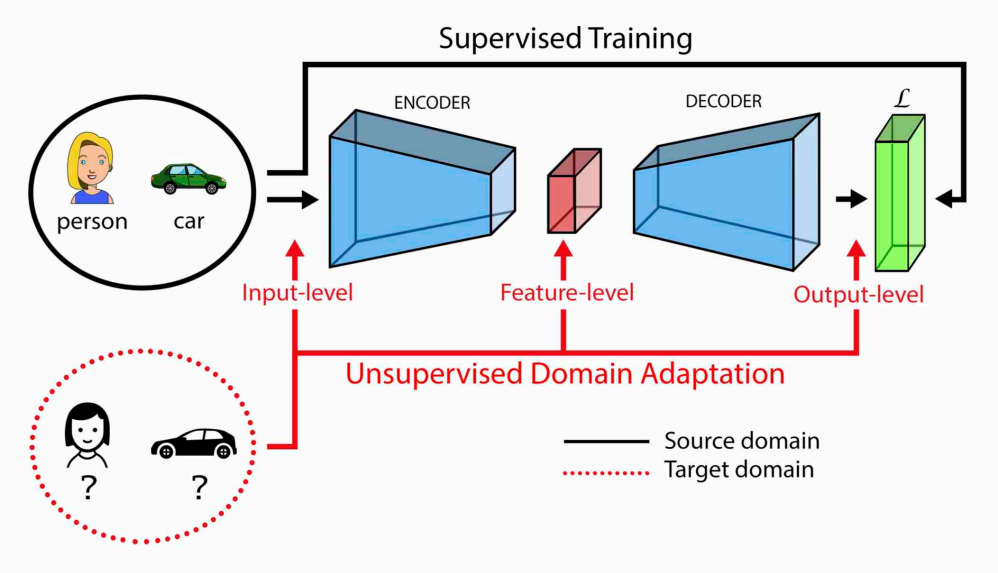


Figure 7.1: Graphical representation of the unsupervised domain adaptation process. A task-loss $\mathcal{L}$ (*e.g.*, a cross-entropy loss) is used for a supervised training stage on the source domain using the semantic annotations. Unsupervised adaptation to target data without labels can be performed at different levels (*e.g.*, input, features or output) with different strategies.

ing (CL) are emerging to overcome the need for multiple changes in domains and tasks [238–240].

7.1.3 Application Motivations

There exists a large number of applications that may significantly benefit from UDA. In general, each application focuses on a very peculiar setting with images taken with a specific camera and a particular environment to solve a prefixed task. The first and easiest solution is to get as much labeled data as possible for the specific problem, but, as already mentioned, this is unfortunately very time consuming and expensive, thus making it unfeasible in many real-world contexts. On the other hand, large and publicly available labeled datasets typically contain generic data and their direct use in specific applications does not grant good performance in the relevant application-specific domain. A second solution would be to transfer source knowledge acquired on a broader scenario and adapt it to the specific setup being targeted. Such context, for example, is fairly common in industrial applications.

An example application is face recognition, which represents a challenging problem that has been actively researched for many years. Current models for face recognition perform very well when training and testing images are acquired under controlled conditions. However, their accuracy quickly degrades when the test images contain variations that are not present in the training images [236]. For instance, these variations could be changed in pose, illumination or viewpoint, and depending on the composition of training and test sets, this can be regarded as a domain adaptation problem [236, 241].

Another straightforward application lies in object recognition, where one may be interested in adapting object detection capabilities from a typically larger set to a specific small-size dataset [242].

Furthermore, the recent improvements in the computer graphics field allowed the production of a large amount of synthetic data for many vision-related tasks. This allows to easily obtain

Figure 7.2: Autonomous cars, industry robots and home assistant robots are just some of the possible real-world applications of Unsupervised Domain Adaptation (UDA) in semantic segmentation. (The images are modified version of pictures obtained with kind permission from Shutterstock, Inc. The original versions have been created (from left to right) by Scharfsinn, Monopoly919 and PaO_Studio).

large training sets but on the other side the domain shift between synthetic and real-world data needs to be addressed. In this field, one of the most interesting applications is found in autonomous vehicles scenarios, where accurate understanding of the surrounding environment is crucial for a safe navigation in an urban context. At the same time, synthetic urban scenes provided with automatically-generated annotations can be easily accessed from highly realistic computer graphic tools [243]. This allows to bypass the time-consuming and highly expensive manual labeling required for real-world training data, but in turn demands for domain adaptation techniques to safely bridge the statistical discrepancy between synthetic and real images. The synthetic to real adaptation for semantic segmentation of urban scenes will be further discussed in Section 7.4.

In Figure 7.2 we show three typical scenarios in which UDA for semantic segmentation could be highly valuable: namely, autonomous vehicles, industrial automation and domestic robots.

7.1.4 Outline

In this part of the thesis, we mainly focus on analyzing and discussing deep UDA methods in semantic segmentation. Recently, there has been a large number of studies related to this task. However, the motivating ideas behind these methods are different. To connect the existing work and hence to better understand the problem, we organize the current literature into some categories. We hope to provide a useful resource for the research of UDA in semantic segmentation.

The rest of this chapter is organized as follows: in Section 7.2 a concise and precise formulation of UDA for semantic segmentation is given outlining the various stages at which the adaptation process may occur. Then, in Section 7.3 we give an overview of the state-of-the-art literature on the topic. We start from precursor techniques with weak supervision and then we propose a categorization based on the techniques employed to align the source and target distributions. In addition, we overview some new research directions considering more relaxed assumptions over target dataset properties, for example, dealing with the detection and classification of unseen semantic categories in target samples. In Section 7.4 we introduce a case study of synthetic to real unsupervised adaptation for semantic understanding of road scenes and we give an overview of the results of existing methods grouped by network architecture and evaluation scenario. In Section 7.5 we conclude our review with some final considerations on the different adaptation techniques.

7.2 Unsupervised Domain Adaptation for Semantic Segmentation

7.2.1 Problem Formulation

Domain Adaptation (DA) is a special case of transfer learning, called Transductive Transfer Learning, in which the source and target tasks coincide ($\mathcal{T}_S = \mathcal{T}_T$), whereas the discrepancy lies in the domain difference ($\mathcal{D}_S \neq \mathcal{D}_T$). In addition, domain adaptation is commonly intended in a homogeneous fashion, when the domain shift happens at a statistical level ($P(X_S, Y_S) \neq P(X_T, Y_T)$) rather than being due to distinct input spaces ($\mathcal{X}_S$ and $\mathcal{X}_T$ belong to the same semantic domain, *e.g.*, urban scene images) [230].

Image classification and image segmentation can both be attributed to the problem of finding a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from the domain space $\mathcal{X}$ of input images to the label space $\mathcal{Y}$, that contains, respectively, the classification tags or the semantic maps. From a mathematical point of view, it is possible to suppose that all real-world labeled images $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are drawn from an underlying, fixed and unknown probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The search of the function h should be limited to a predefined function space $\mathcal{H}$, called hypothesis class, chosen based on the prior knowledge on the problem. In a supervised setting, a dataset of i.i.d. samples from $\mathcal{D}$ is used by the learner to find the best mapping $h \in \mathcal{H}$ (*i.e.*, the solution that minimizes a cost function over the training set). On the other hand, in DA, two different and related distributions over $\mathcal{X} \times \mathcal{Y}$, namely a source distribution $\mathcal{D}_S$ and a target distribution $\mathcal{D}_T$, are considered. A source domain training set $\mathcal{S}$ is sampled from $\mathcal{D}_S$ and a target domain training set $\mathcal{T}$ is sampled from $\mathcal{D}_T$ or from its marginal distribution over $\mathcal{X}$. The main purpose of DA is to use labeled i.i.d. samples from source domain $\mathcal{S}$ and labeled, or unlabeled, or a mixture of both, i.i.d. samples of the target domain $\mathcal{T}$ to find a hypothesis $h \in \mathcal{H}$ that performs well on the target domain $\mathcal{T}$. The DA task is supervised if labels in the target domain are available for all samples; it is semi-supervised if labels are available for just some samples; or it is unsupervised if the target samples are completely unlabeled (*i.e.*, they are drawn from the marginal distribution of $\mathcal{D}_T$ over $\mathcal{X}$). Domain adaptation can be subdivided even further based on the categories (*i.e.*, classes or labels) of the source (C_S) and target (C_T) domains, and on the categories considered in the learning process (C_L):

- *Closed Set DA*: it corresponds to the homogeneous case, where semantic classes are completely shared between source and target domains ($\mathcal{C}_S = \mathcal{C}_T$).
- *Partial DA*: in this setup there exist some source classes that do not appear in the target domain ($\mathcal{C}_S \supset \mathcal{C}_T$).
- *Open Set DA*: conversely to partial DA, here the presence of some target private classes is admitted, for which no training examples in the source domain are available ($\mathcal{C}_S \subset \mathcal{C}_T$).
- *Open-Partial Set DA*: the source and target domains include separate sets of semantic classes [244], with a subset of those in common ($\mathcal{C}_S \neq \mathcal{C}_T$, $\mathcal{C}_S \cap \mathcal{C}_T \neq \varnothing$). However, elements belonging to the class subset exclusive to the target domain have only to be acknowledged as not part of the shared classes.
- *Boundless DA*: this setup is very similar to the open set one, but objects of target private classes must be explicitly classified rather than only be associated to a general unknown target class. This setting has been recently introduced [245] and it represents the most ambitious one, since it admits complete unawareness beforehand about semantic content of target data.

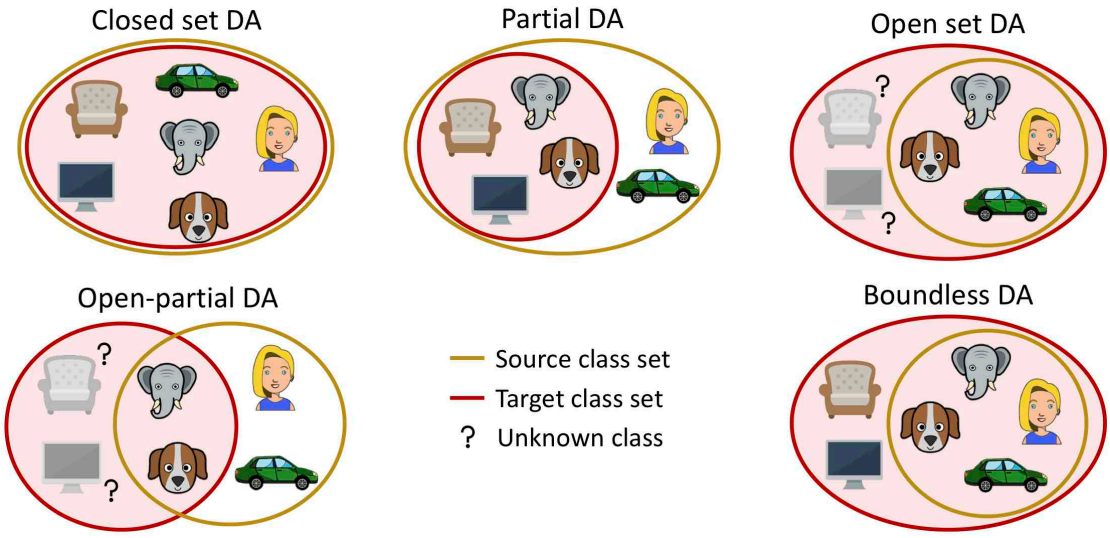


Figure 7.3: Different settings for domain adaptation, according to how source and target class sets are related.

It is important to remark that in Open Set DA, usually, the categories of the target set that do not belong also to the source domain are learned by the model as an *unknown* additional class, while in Boundless DA [245] they are learned individually. An overview of the aforementioned classification is given in Figure 7.3.

In the following sections, the focus will be placed on the *standard* most-diffused closed set adaptation, as, up to now, this is by far the most explored setup.

According to the degree of annotations availability in the target domain, the adaptation problem is subject to a further categorization, ranging from the full or partially annotated *supervised* or *semi-supervised* settings to the completely label deprived *unsupervised* scenario. In particular, Unsupervised Domain Adaptation (UDA) will be discussed, as it has recently witnessed an increase in popularity, especially in relation to the semantic segmentation task, and it involves many practical applications. More specifically, it is assumed that a set of labeled source data $\{x_i^s, y_i^s\}$ drawn accordingly to the source joint distribution over $\mathcal{X}_S \times \mathcal{Y}_S$ is provided, paired with a set of unlabeled samples $\{x_i^t\}$, retrieved from a distinct target marginal distribution over $\mathcal{X}_T$. The objective is to discover a predictive function correctly modeling the task input-label relation in the target domain, while knowledge on the chosen task can be extracted only from source labeled samples.

Furthermore, for standard domain adaptation techniques to work, source and target domains should be somehow related, meaning that they should share task-relevant content, while low-level attributes may differ. This scenario is commonly referred to as *one-step* DA, as knowledge transferring happens directly across source and target data without intermediate stages.

7.2.2 UDA in Semantic Segmentation: Adaptation Focuses

As previously discussed, behind the performance degradation suffered by deep prediction models applied on new target environments lies the covariate shift phenomenon affecting source and target input data samples. For this reason, most of domain adaptation research builds upon bridging the statistical gap between domain distributions, in order for the prediction model to yield satisfactory results whenever those distributions are matched.

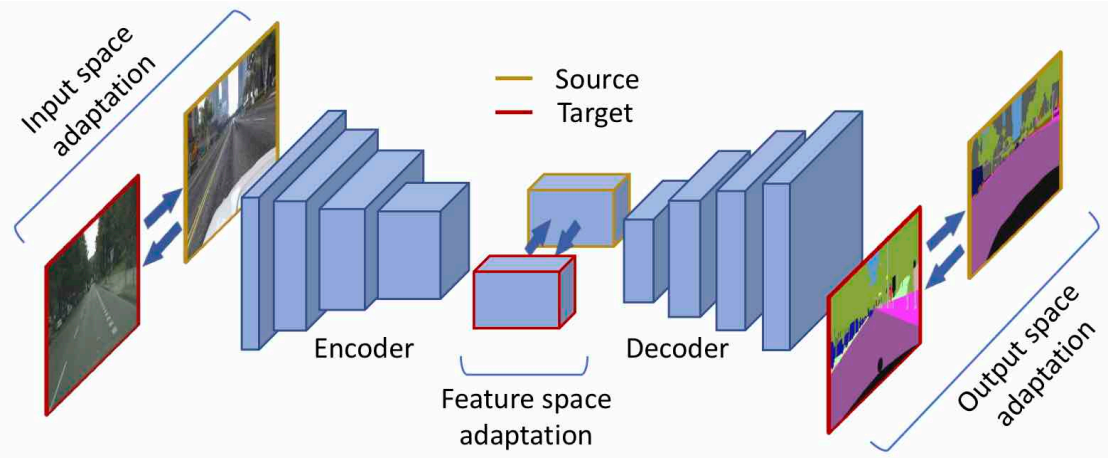


Figure 7.4: General scheme of an auto-encoder network for semantic segmentation highlighting the different network stages on which domain adaptation strategies can be applied, from the input image space up to intermediate or output network activations.

Various strategies have been explored to achieve the statistical matching, which will be thoroughly discussed in Section 7.3. A more general categorization of domain adaptation techniques, however, can be inferred, according to where in the employed semantic segmentation model the statistical discrepancy happens to be addressed. In particular, different data representations could be subject to adaptation, from the bare images prior to classification up to intermediate and output network activations (Figure 7.4). In the following, a description of the main ideas behind adaptation approaches will be provided, grouped by where the adaptation effort is focused.

7.2.2.1 Input Level Adaptation

A first strategy is to perform adaptation at the input level, directly on images before they are fed to the segmentation network (as shown in the leftmost part of Figure 7.4). The idea is to force data samples from either domain to reach an uniform visual appearance, meaning that they not only have to carry high-level semantic similarity, but their low-level statistical discrepancy should be matched as well. This because low-level domain dependent attributes, even though they do not define the semantic content of the input image, can still be captured by the prediction model, thus leading to incorrect predictions when a domain change alters them. A clear example of this is the synthetic to real adaptation; although it may be quite realistic, synthetic data can mimic real-word properties up to a certain extent. Thus, it is usually possible to find synthetic peculiar traits, however small, which can undermine the efficacy of a model trained on synthetic data in a real-world environment.

The common approach to address domain adaptation at the input level is to map the data to a new image space, where the projected source (or target) samples carry an enhanced perceptual resemblance to target (or source) ones. This is normally achieved with the help of style-transfer techniques, whose objective is turned into matching source and target marginal distributions in the image space. By feeding in input supervised data from the new domain-invariant space to the segmentation network, the predictor should now able to retain consistent results across domains.

An upside of this approach is its complete independence with respect to the segmentation

network currently in use that does not require any modification. This, however, comes with a cost, which is that, in its vanilla scheme without any extra regularizing factors, marginal alignment may be performed without the class-conditional distributions being simultaneously matched. In other words, it may be possible to end up with domain invariant representations, which yet lack the semantic coherence with the original data crucial to solve the segmentation task. To get past this problem, multiple solutions have been proposed to achieve semantically consistent image translations, for example through image reconstruction constraints or additional loss components enforcing the uniformity of segmentation predictions.

We will apply an input-level adaptation approach [30] in Chapter 9.

7.2.2.2 Feature Level Adaptation

An alternative approach is to focus the adaptation on feature representations, pursuing a distribution alignment of network latent embeddings, which are normally retrieved from the encoder output in the commonly employed auto-encoder architecture (even if adaptation at other network stages has also been employed). The primary objective is to build a domain invariant latent space, in which features extracted either from source or target input images observe the same distribution. In the end, learning solely from supervision on source representations should result in a good performance also on the target domain, as shared classification in the adapted latent space should be jointly effective on both source and target representations when distributed alike.

In the context of semantic segmentation, the feature space retains significant complexity due to its high-dimensionality, which is necessary for the prediction model to simultaneously capture global semantic clues, while attaining pixel-level accuracy. In addition, as for the input level adaptation, a semantically unaware alignment of marginal distributions (*e.g.*, standard adversarial adaptation) does not guarantee that the joint input-label distributions are matching, since no information can be derived from unlabeled target samples about the target joint distribution. For these reasons, many feature level adaptation techniques that have been successfully devised for image classification do not easily extend to the dense segmentation task, and in general require careful tuning and further regularization.

We will introduce some novel feature-level adaptation approaches [30–33] in Chapter 9.

7.2.2.3 Output Level Adaptation

Finally, the last class of domain adaptation techniques exploits a cross-domain distribution alignment over the network output, *i.e.*, typically the output per-class probability space. Not only prediction probability maps have proven to retain sufficient complexity and richness of semantic information, but they also span a low-dimensional space over which statistical alignment happens to be achieved much more effectively, for example by the domain adversarial strategy. In addition, source knowledge can be indirectly translated over the unlabeled target domain by resorting to some form of self-taught supervision extracted from target prediction maps, whose careful introduction in the learning process to support the standard source supervision may result in an effective cross-domain adaptation of the network performance. Source priors derived from label distribution have proven to provide an useful regularization to the learning process as well, since they usually identify high-level semantic properties shared across domains.

We will focus on output-level adaptation in Chapter 8, where our contributions are proposed [26–28].

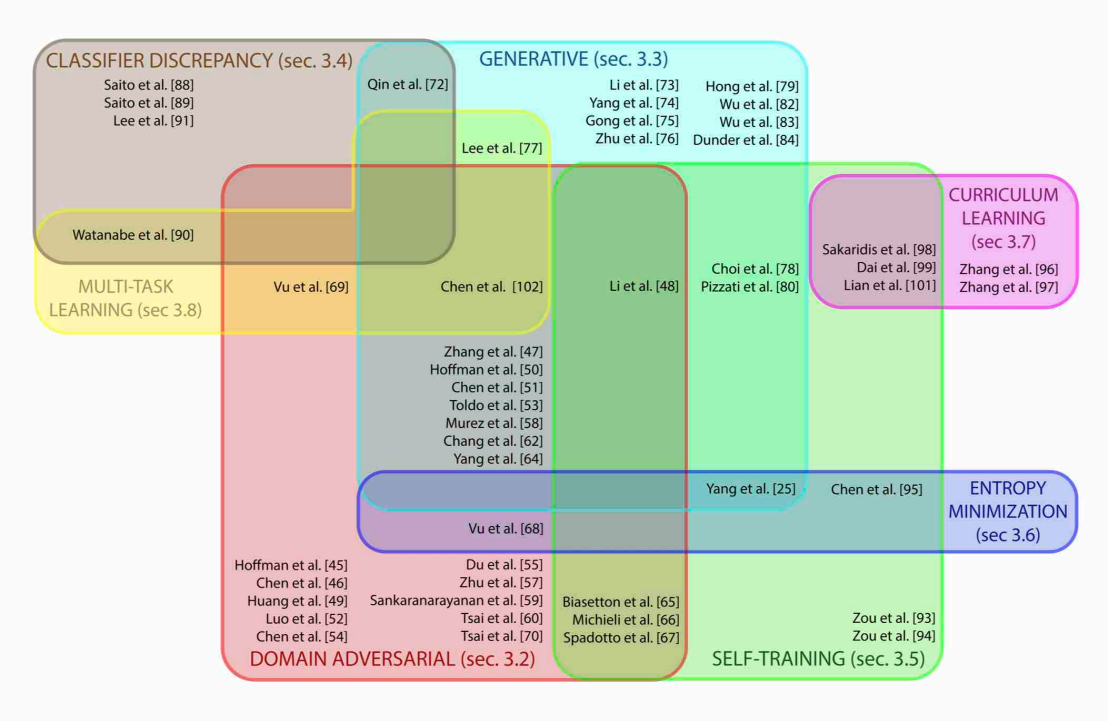


Figure 7.5: Venn diagram of the most popular UDA strategies for semantic segmentation. Each method falls in the set representing the adaptation techniques used.

7.2.2.4 Ad-Hoc Level Adaptation

In addition to the aforementioned techniques, other works resort to a distribution alignment over ad-hoc spaces upon network activations. Such methods aim at better capturing high-level patterns essential to solve the segmentation task, and ultimately achieve an improved match of source and target embeddings, thanks to gradients flowing back through the segmentation network at different levels. Hence, the adaptation is not only restricted to a particular network level, *i.e.*, at the end of the feature extraction network, but it is achieved at intermediate levels as well.

7.3 Review of Unsupervised Domain Adaptation Strategies

This section reviews the most relevant approaches for Unsupervised Domain Adaptation in semantic segmentation. We start this section by presenting some weakly- and semi- supervised learning methods for semantic segmentation. Those are not purely UDA approaches since they require some minimal supervision with annotations on typically simpler tasks, but have represented the starting point in dealing with the domain adaptation problem. Then, we grouped UDA approaches into seven main categories, as shown by the visual overview in Figure 7.5. Domain adversarial discriminative approaches (Section 7.3.2) learn to produce data with a statistical distribution similar to that of training samples via adversarial learning schemes. Generative-based approaches (Section 7.3.3) typically use generative networks to translate data between domains in order to produce a target-like training set from source data, or alternatively to translate the

source data into a representation closer to target domain characteristics that can then be fed to the network. Classifier discrepancy approaches in Section 7.3.4 resort to multiple dense classifiers on top of a single encoder to capture less adapted target representations and, in turn, encourage an improved alignment of cross-domain features far from decision boundaries via an adversarial-like strategy. Self-training approaches in Section 7.3.5 propose to produce some form of pseudo-label (typically using some confidence estimation schemes to select the most reliable predictions) based on the current estimate to automatically guide the learning process (self-supervising it). Entropy minimization methods in Section 7.3.6 aim at minimizing the entropy of target output probability maps to mimic the over-confident behavior of source predictions, thus promoting well-clustered target feature representations. Curriculum learning approaches in Section 7.3.7 tackle one or more easy tasks first, in order to infer some necessary properties about the target domain (*e.g.*, global label distributions) and then train the segmentation network such that the predictions in the target domain follow those inferred properties. Multi-tasking methods in Section 7.3.8 solve multiple tasks simultaneously to improve the extraction of invariant features representation. Latent-level regularization in Section 7.3.9 approaches align feature representations across the datasets promoting domain invariance and reducing overlapping active channels. Finally, in Section 7.3.10 we conclude our digression with some considerations about recent interesting research directions to be further expanded in the future.

For each set of techniques, some works whose proposed adaptation solutions can be associated to that class will be presented. However, it should be stressed that most of the domain adaptation frameworks recently introduced resort to a combination of multiple techniques to improve performance.

7.3.1 Weakly- and Semi-Supervised Learning

As earlier semantic segmentation works started from image classification techniques, also the domain adaptation task was originally tackled in the classification field, in that the first DA approaches for semantic segmentation have been developed by adapting DA methods for classification. However, approaches directly targeting the semantic segmentation task started to appear soon, taking into account the specific properties of the spatial components (completely missing in the classification methods) and of the dense (pixel-wise) task. At the same time, unsupervised domain adaptation was historically preceded by techniques with weak or partial supervision, which are the focus of this section.

As we already mentioned, the training of a deep learning model for semantic segmentation requires a large amount of data with pixel-level semantic labels that are very expensive, difficult, frustrating and time-consuming to acquire. Such problem is not so relevant for other computer vision tasks like image classification and object detection because image-level tags or bounding boxes (that in this context are called *weak* labels) are much simpler to obtain and large annotated collections are available. This is the motivation behind many works that propose to use just weakly labeled samples to train a model in the segmentation task (weakly-supervised learning) [246–248] or to use a mixture of many weakly labeled samples and few samples with the more expensive pixel-level semantic map (semi-supervised learning) [249, 250].

A first approach to solve the problem is to cast the weakly supervised semantic segmentation as a Multiple Instance Learning problem, as shown in [251, 252]. The Semantic Texton Forest (STF) traditional feature-based approach has been used as base framework and an algorithm to estimate unobserved pixel label probabilities from image label probabilities has been introduced. Then, the structure of the STF has been improved through a new algorithm that uses a geometric context estimation task as regularizer in a multi-task learning framework.

Another strategy, proposed in [253], is to implement an Expectation-Maximization (EM)

method to train a deep network for the semantic segmentation task in a weakly- and semi-supervised setup. The algorithm alternates between estimating the pixel-level annotations (constrained on the weak annotations) and the optimization of the segmentation network itself. In [254], Constrained CNNs (CCNNs) have been introduced as a framework to incorporate weak supervision into the training. Linear constraints are added in the output space to describe the existence and expected distribution of labels from image level tags and a new loss function is introduced to optimize the set of constraints.

In [255], a simple to complex framework has been introduced for weakly-supervised semantic segmentation. In the paper, a distinction is made between simple and complex images: the former include a single object of just one category in the foreground and a clean background, the latter can have multiple objects of multiple categories with a cluttered background. First, salient object detection techniques are used to compute semantic maps from weak-annotated simple images and, then, starting from these, three different networks are trained, sequentially, in order to gradually enable the segmentation of complex images.

In [256], a semi-supervised approach is presented, in which the architecture is composed of three main structures: a classification network, a segmentation network and some bridging layers that interconnect the two networks. The proposed training is decoupled: first, the classification network is trained with weakly-annotated samples, then, the bridging layers and the segmentation network are jointly trained with the strong-annotated samples. The input image is first fed to the classification network, then the bridging layers extract from an intermediate layer of the classification network a class-specific activation map that is used as input for the segmentation network. In this way, it has been possible to reduce the number of parameters of the segmentation network and to make a training with just few semantic-annotated samples possible. In fact, relevant labels and spatial information are captured from the classification network and refined by the bridging layers and the task of the segmentation network is widely simplified.

In [257], an iterative procedure has been proposed to train a segmentation network just with bounding-boxes annotated samples. First, region proposal methods are used to generate many candidate segmentation masks (that are fixed throughout the training) for each image. An overlapping objective function is defined to pick the candidate mask that overlaps the ground truth bounding box as much as possible with the correct label. At every iterative step, one candidate mask is selected for each bounding box and then the resulting semantic labels are used to train the segmentation network. The outputs of the segmentation network are then used to improve the choice of the candidate labels for the next step through a feedback channel. After every iteration, the selected candidate labels and the segmentation network outputs both improve together.

Generative adversarial networks have proven to be effective in this field starting from [258], where the discriminator network is modified to accomplish the task of semantic segmentation. The discriminator assigns to every pixel of the input image either a label of one of the semantic classes or a fake label. The discriminator is trained with fake (generated) data, unlabeled data for regularization purposes, and with labeled data with pixel-level semantic maps. Another proposed solution is to employ conditional Generative Adversarial Networks (GANs) and incorporate weak image-level annotation both at the generator and at the discriminator inputs in a weakly-supervised setup.

Many approaches of self-supervised learning have been proposed starting from [259]. The common rationale is the exploitation of inferred pixel-level activations as pseudo ground-truth in order to obtain more accurate pixel-level segmentation maps. In [260], an image classification network with classification activation maps has been used. The authors highlight how the discriminative regions, using that method, are small and sparse and they propose to use them as seed cues. Then, the regions are expanded to neighboring pixels with similar features (for ex-

ample color, texture or deep features) with a classical Seeded Region Growing (SRG) algorithm to obtain accurate pixel-level labels that are used to train a segmentation network. The output of the segmentation network is used by the SRG algorithm to compute the similarity between the seed and the adjacent pixels. So, at every iteration, the segmentation network and the dynamic labels computed with SRG improve together. A similar approach that introduces a new adversarial erasing method for localizing and expanding object regions progressively is presented in [261]. Other self-learning based techniques are presented in [262–264].

A more general framework to transfer knowledge across tasks and domains is presented in [265]. Assuming to have two tasks and two domains, the proposed method works in four steps: (1) a single task network is trained on samples of both domains to solve the first task, in order to find a common feature representation for the domains; (2) a second network is trained to solve the second task just on the first domain; (3) a third network is trained on the first domain to map deep features suitable for the first task into features to be used for the second task; (4) finally, the third network is used to solve the second task on the second domain. This framework enables to adapt from a synthetic domain to a real one for the image segmentation task using depth maps of both domains. The depth maps can be considered weak annotations with respect to semantic maps because their acquisition is easier thanks to depth cameras and 3D scanners.

7.3.2 Domain Discriminative

Domain dissimilarity between source and target distributions has been initially tackled with traditional methods. Some works refer to the Maximum Mean Discrepancy (MMD), such as [237, 266, 267]. Tzeng *et al.* [266] introduce an adaptation layer and a domain confusion loss in the standard CNN architecture to learn domain invariant representations. In [237, 267] the degradation in transferability of application-specific hidden representations is tackled by matching mean domain embeddings in a new space. A different approach relies on correlation alignment, taking into account second order statistic properties of the dataset [268, 269]. A key factor for the majority of these works is that the deep network adaptation is performed end-to-end by assisting the task loss with a supplementary adaptation objective. Furthermore, no restriction to a specific network architecture is assumed.

Adversarial Learning: Adversarial learning has been introduced in the form of Generative Adversarial Networks (GANs) [108] to address a generative objective (*i.e.*, generating *fake* images similar to real-world ones). Solving the generative task can be thought as seeking the evaluation of the unknown probability distribution from which the training data has been generated. In the generative context, the introduction of adversarial learning has been ground-breaking, as explicit modeling of the underlying target distribution is not required and, more importantly, no specific objective is needed to train the model. The learning process builds upon a min-max game, where a generator network is progressively guided by a discriminator network to produce realistic samples. In the adversarial scheme, a generator has to learn to produce data with the same statistical distribution of training samples. To do so, it is paired with a discriminator, which has the goal of understanding whether input data comes from the original set or, instead, it has been generated. At the same time, the generator is optimized to fool the discriminator by producing samples that resemble the original ones. In the end, the statistics of generated data should match that of the training set. The GAN model is capable of learning a structured loss in the form of a learnable discriminator, which guides the generative network in its optimization procedure. For this reason, the objective function can be thought as automatically adapting to the specific context, removing, in fact, the necessity to manually design complex losses. Therefore, the adversarial learning scheme introduced in [108] can be extended under careful

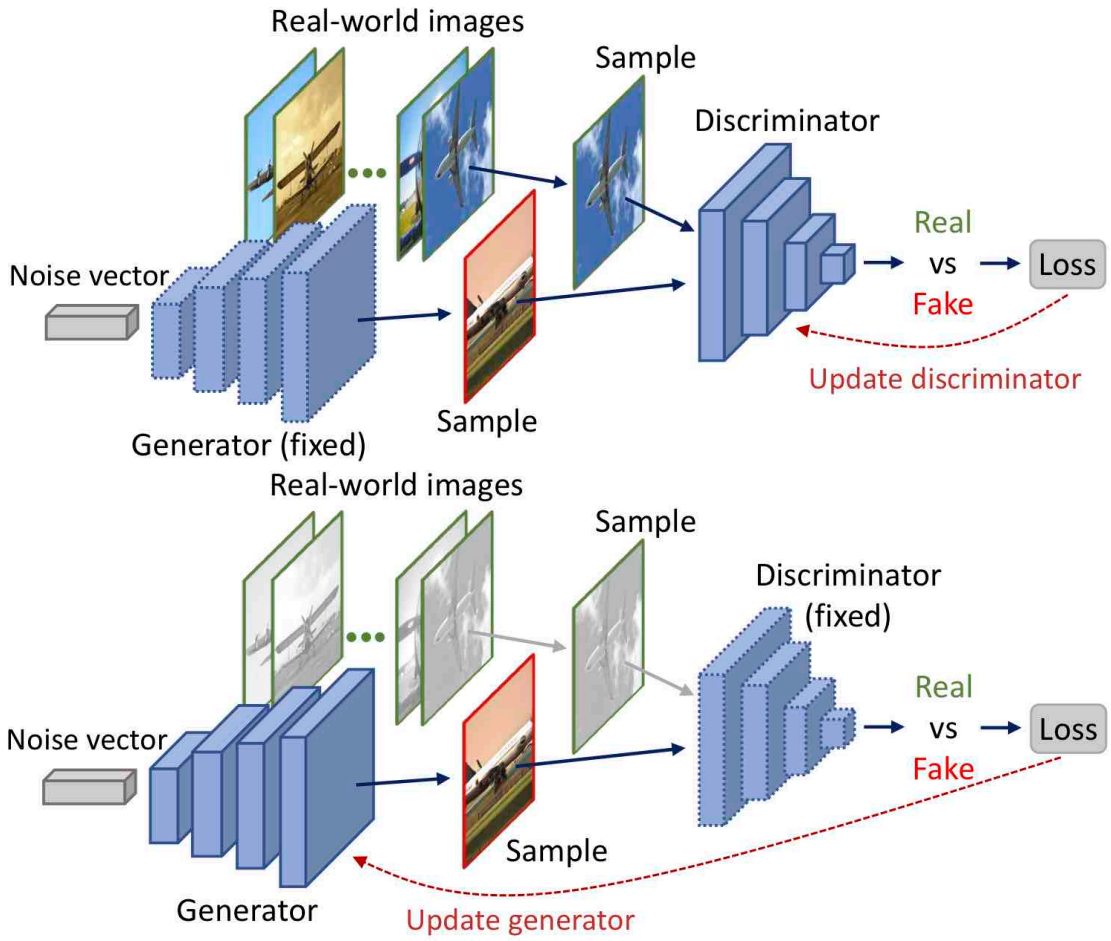


Figure 7.6: Training of a generative adversarial network. Update step of the discriminator (top) and of the generator (bottom).

adjustments to address multiple tasks that would normally require different types of application-specific objectives.

In [108], the discriminator is a binary classifier whose goal is to discern between the original training data and the data produced by the generator. The generator is instead a generative model that takes in input random noise (or some conditioning data in more recent variations of the approach) and produces data (i.e., images in the setting of interest) resembling the ones in the training set. It aims at constantly improving the realism of its output samples to fool the discriminative action of its opponent, and this is achieved by using a loss function whose minimization in turn maximizes the errors of the discriminator. The model is trained by alternating a discriminator training step, aiming at maximizing its accuracy, and a generator optimization phase with the opposite target (see Figure 7.6). If correctly carried out, the adversarial competition should result in a statistical distribution of generated data that fully matches the training set one, meaning that original and generated data should be statistically indistinguishable. In addition, the discriminator should be able to both capture and express a measure of statistical discrepancy in the form of a structured learnable loss. Therefore, the objective function can be thought as being jointly learned and optimized in the adversarial process, allowing it to adapt

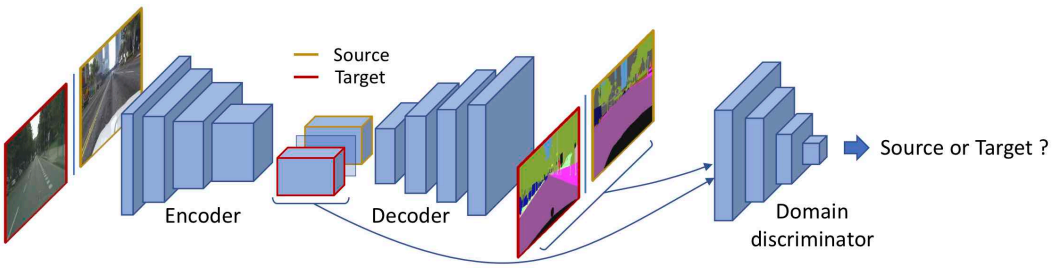


Figure 7.7: Graphical representation of the standard adversarial adaptation strategy. A domain discrimination captures the statistical discrepancy between source and target representations (*e.g.*, segmentation network's output or features maps computed from one or the other domain). Its supervisory signal is then exploited to perform domain alignment.

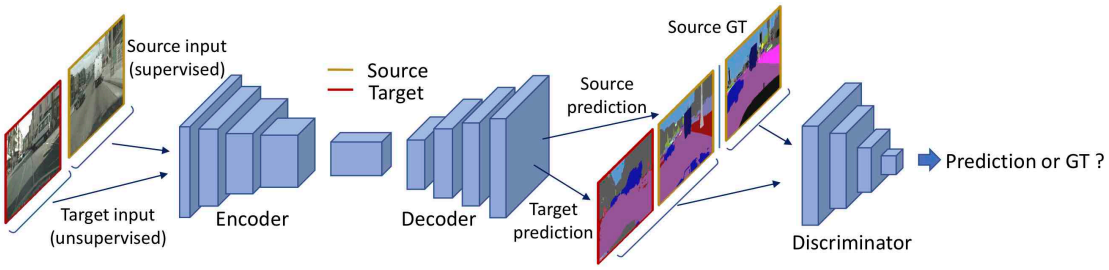


Figure 7.8: Graphical representation of an output adversarial adaptation strategy, where domain alignment is performed indirectly by bridging the distribution gap between source annotation maps and network predictions from either source or target domains.

to the specific context.

Adversarial learning has been successfully extended to the domain adaptation task. The real-fake discriminator is now turned into a domain classifier that is used to drive the adaptation process. Its discriminative action is, in fact, focused on capturing the statistical discrepancy between representations from separate domains, which is responsible for the performance degradation and thus has to be reduced in order to achieve an effective adaptation.

There are two possible targets for the domain classifier. The first is to discriminate between internal or output representations extracted from data in either source or target domains (Figure 7.7). This allows to introduce additional loss terms enforcing the construction of feature or output spaces that are more domain invariant. Alternatively, it is possible to use the discriminator to distinguish between the output of the network (that can correspond both to inputs from the source and from the target domain) and the ground truth segmentation maps (that in the unsupervised setting are only present in the source domain). Since in the adversarial model there is no need to have ground truth data matching the provided samples, this allows to use also the target domain images for which no ground truth is available and to enforce that their predicted segmentation maps have statistical properties similar to the ground truth ones (Figure 7.8). Using these strategies, the standard supervision from the annotated source data is joined by a supervisory signal from the domain discriminator, which pushes the prediction network towards domain invariance, in turn mitigating the intrinsic bias towards the supervised source domain.

Feature Adversarial Adaptation: Aiming at exploiting the statistical matching that can be achieved by the GAN model, adversarial learning has been successfully extended to the

domain adaptation task [270–272]. In particular, the real-fake discriminative network in the original adversarial framework is revisited, turning into a source-target domain classifier. Thus, while the segmentation network is trained with source supervision to achieve discriminability over the semantic segmentation task, the supervisory signal provided by the domain discriminator should guide the predictor to reach domain invariance and reduce the otherwise intrinsic bias towards the source domain. In other words, a measure of domain discrepancy is simultaneously learned and minimized within the adversarial competition.

Although the adversarial adaptation strategy has been originally introduced for the image classification task [270, 271], it has been later extended to image semantic segmentation. Hoffman *et al.* [273] have been the first to address domain adaptation in semantic segmentation, and they resort to an adversarial approach. In particular, they devise a global domain adversarial alignment, based on a domain discriminator taking as input the feature representations from intermediate activations of the fully convolutional segmentation network. In addition, they propose a category specific distribution alignment, which is accomplished by enforcing image-level label distribution constraints on target predictions inferred from source annotations, under the assumption that high-level scene layout is in general shared among source and target images.

Yet, as previously discussed, the global domain alignment of marginal distributions provided by the vanilla domain adversarial scheme may end up in incorrect semantic knowledge transfer across domains, with class-conditional distributions neglected in the learning process. For this reason, to reach an effective adversarial adaptation when dealing with the semantic segmentation task, additional modules should be embedded in the adaptation pipeline.

A possible solution is to integrate adversarial feature alignment in a generative approach (see Section 7.3.3), as done in several works [30, 274–276]. Here the goal is to strengthen the image space adaptation, so that the attribute transferring to match visual appearances of images from different domains is extended inside the feature space. An alternative is to perform category-wise adaptation [277, 278]. The idea is to resort to class-wise adversarial learning, by introducing multiple per-class distinct feature discriminators, that, in principle, should provide a semantically consistent knowledge transfer, which is absent in the standard global adaptation. Finally, with a different perspective it is possible to rely on a reconstruction constraint to enforce domain invariance over latent feature embeddings [279–281]. Adversarial learning in this case is applied over the reconstruction image-level space, to guarantee that feature representations can be projected back to either source or target image spaces without distinction.

As observed, the semantic segmentation task entails a quite complex feature space, due to the high dimensionality of its representations. Thus, to bypass the complexity encompassed in feature space adaptation, a research direction has been to focus the adaptation effort to the segmentation output space [26–28, 282–286]. The low-dimensional output representations, in fact, have been shown to retain enough semantic information for a successful adaptation. In this new output level adversarial scheme, a domain discriminator learns to discover the domain from which segmentation maps are originated. Simultaneously, the segmentation network plays the generative role by providing cross-domain statistically close predictions to fool the discerning action of the domain classifier. While the common solution has been to align source and target output representations [282–286], some works have revisited the standard approach by seeking for an indirect domain alignment [26–28], by forcing predictions from either domain to be distributed as ground-truth source labels (as depicted in Figure 7.8).

Following a similar approach to [273], many works have further resorted to adversarial alignment of network latent embeddings [30, 274–278, 287–290]. As previously discussed, the domain discriminator is able to infer a structured loss to capture global distribution mismatch of cross-domain image representations. However, global alignment of marginal distributions does not necessarily result in class-wise correct semantic knowledge transfer from source to target rep-

resentations. Thus, adversarial learning is commonly employed in more complex frameworks working also on the internal feature representations of the network, comprising multiple complementary modules to achieve a more effective adaptation. For example, Chen *et al.* [287] use an additional target guided distillation loss by matching network activations from target inputs during the training phase with those from a pre-trained version on the ImageNet dataset [121]. In this way, they argue that overfitting to source data is decreased. Moreover, the feature adversarial adaptation is enforced independently over different spatial regions of the input image, thus exploiting the underlying spatial structure of input scenes. Zhang *et al.* [288], instead, boost the feature-level adaptation performance by providing the domain discriminator with an Atrous Spatial Pyramid Pooling (ASPP) module [5] to capture multi-scale representations. More recently, Luo *et al.* [290] propose a significance-aware information bottleneck (SIB) to filter out task-independent information encoded inside feature representations, so that, when enforcing adversarial adaptation, only domain invariant discriminative cues are preserved. They also introduce a significance-aware module to help the prediction of less frequent classes, which may be penalized by the information bottleneck in its original form.

Another group of research [30, 274–276] combines a generative approach (which will be extensively discussed in the following Section 7.3.3), with the adversarial feature alignment. In particular, source and target marginal distributions are matched in the input image space by a source-to-target image-to-image translation function, and then cross-domain latent representations are further brought closer by matching source original and target-like source embeddings in a domain adversarial fashion.

To accomplish category-wise adaptation, some works [277, 278] revisit the original approach of Hoffman *et al.* [273] by assisting the global distribution alignment with class-wise adversarial learning. Chen *et al.* [277] propose to exploit multiple feature discriminators (one for each class), so that negative transfer among different classes in the domain bridging process should be effectively avoided. In addition, due to lack of ground-truth maps, they use grid-level soft pseudo labels from network predictions to compute the target adversarial loss. Recently, Du *et al.* [278] proposed a similar class-wise adversarial technique, which is improved by imposing independence during the optimization of the multiple discriminators. They argue that soft labels lead to incorrect adaptation on class boundaries, where different class discriminators may provide their guidance simultaneously. Finally, they devise an additional module to adaptively re-weight the contribution of each class component in the adversarial loss, in order to avoid the inherent dominance of classes with higher prediction probability, which turns out to be more easily well-adapted across the domains.

Different from the aforementioned techniques, other works [279–281] seek for domain alignment inside the feature space by applying a reconstruction constraint to ensure that latent embeddings possess enough information to recover the input images from which they have been extracted. To this end, adversarial learning is applied on the reconstruction image-level space. To achieve cross-domain feature distribution alignment, the feature extractor is trained to yield latent representations that can be projected back to both source and target image spaces indistinctly. In these frameworks the backbone encoder of the segmentation network plays a min-max game against the domain discriminator. The encoder, indeed, tries to fool the discriminator on the actual originating feature's domain, by looking at the corresponding reconstructed images projected back into the image space. In other words, the objective is to learn source (target) features that can successfully generate target-like (source-like) images to promote domain invariance of those representations.

Output Adversarial Adaptation: To avoid the complexity of high-dimensional feature space adaptation, a different line of works [26–28, 282–286] resort to adversarial adaptation on the low-dimensional output space spanned by the segmentation network, which is still expected

to encode enough semantic information to allow effective adaptation. A domain discriminator is provided with prediction maps from source and target inputs and it is optimized to discern the domain they originate from. Conversely, the segmentation network has to fool it by aligning the distribution of predicted dense labels across domains. Tsai *et al.* [282] are the first to propose this type of adaptation: in order to improve the signal flow from the adversarial competition through the segmentation network, they deploy multiple dense classification modules at different depths upon which as many output-level discriminators are applied.
Following the technique proposed in [282], other works adopt the output space adversarial adaptation in combination with additional modules. For example, Chen *et al.* [283] combine semantic segmentation and depth estimation to boost the adaptation performance. In particular, they provide the domain discriminator with segmentation and depth prediction maps jointly, in order to fully exploit the strong correlation between the two visual tasks. Moreover, Luo *et al.* [285] enhances the adversarial scheme by a co-training strategy that highlights regions of the input image with high prediction confidence. In this way, the adversarial loss can be effectively tuned by balancing the contribution of each spatial unit, so that more focus is directed towards less adapted areas.

Other works [26–28] revisit the adversarial output-level approach. In particular, they utilize a discriminator network that has to distinguish between source ground-truth maps and generated semantic predictions from either source and target data. In doing so, the cross-domain statistical alignment is not directly performed, but forcing the segmentation network output to be distributed as ground-truth labels for both source and target inputs leads to an indirect yet effective alignment between the two domains.

Recently, new approaches [151, 291, 292] have been proposed based on the extraction of meaningful patterns from the segmentation output space to be exploited in the adaptation process. This is done to explicitly guide the domain discriminator towards a more functional and significant insight of source and target representations, and thus to ultimately achieve a better alignment.

On this regard, Vu *et al.* [151, 291] devise an entropy-minimization strategy (which will be described more in detail in Section 7.3.6) to promote more confident target predictions. They propose an indirect approach relying on the adversarial alignment of the statistics of self-information maps computed on top of source and target predictions. In particular, a domain discriminator has to detect whether a weighted self-information map comes from a source or a target prediction, whereas the segmentation network, trying to deceive the discriminator, is forced to produce low-entropy target maps as to mimic source confident ones. This process effectively pushes decision boundaries away from high-density regions in the representation space.

With a different approach, Tsai *et al.* [292] construct a clustered space over the output prediction space by adding a patch clustering module that discovers patch-wise modes on segmentation maps. First, the module is trained, while supervised, on source data by leveraging the available annotations, then it is exploited to achieve a patch-wise distribution alignment by enforcing adversarial cross-domain adaptation between its clustered source and target representations. The idea behind this approach is to capture high-level structured patterns, which are essential to solving the semantic segmentation task, to be provided to the domain discriminator for an improved domain statistical alignment. Thus, the achieved domain uniformity on a patch-level should ensure, in principle, that the segmentation task can be effectively solved also in the target domain.

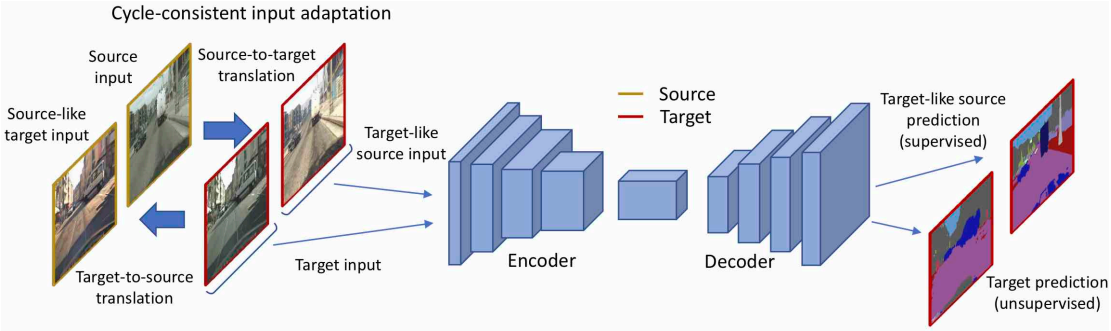


Figure 7.9: Overview of the generative-based adaptation approach built upon cycle-consistent image-to-image translation. In particular, source translated input images are exploited as a form of target-like artificial supervision during the learning process.

7.3.3 Generative-based Approaches

Unsupervised image-to-image translation is a class of generative techniques where the objective is to learn a function that maps images across domains, relying solely on the supervision provided by unpaired training data sampled from the considered domains. The idea is to extract characteristics peculiar to a specific set of images and transfer those properties to a different data collection. In a more formal definition, the image-to-image translation task aims at discovering a joint distribution of images from different domains. Notice that, since the problem is, in fact, ill-posed, as an infinite set of joint distributions can be inferred from the marginal ones, appropriate constraints must be applied to obtain acceptable solutions.

Image-to-image translation can be effectively exploited in domain adaptation: discovering the conditional distribution of the target set with respect to the source one, should allow, in principle, to bridge the statistical gap between source and target pixel-level statistics, thus removing the original covariate shift responsible for the classifier performance drop. The goal, in fact, is to transfer visual attributes from the target domain to the source one, while preserving source semantic information. Following this idea, many works have proposed an input-level adaptation strategy based on a generative module that translates images between source and target domains. Despite the wide range of different approaches, all these works share the same idea of achieving a form of domain invariance in terms of visual appearance, by mitigating the cross-domain discrepancy in image layout and structure. This allows to learn a segmentation network on translated source domain data (that should have a target-like statistical distribution) allowing to make use of source annotations.

Data augmentation techniques can be used to improve the generalization capabilities: some works propose to pre-process the synthetic images, *i.e.*, the existing labeled data, to reduce the inherent discrepancy between real and synthetic domain distributions mainly using generative models based on Generative Adversarial Networks (GANs) [293–297]. Thus, these augmented labeled data are used to train the segmentation network to work in a more reliable way on the real domain.

A considerable amount of research [30,274–276,280,298–303] has been resorting to the successful CycleGAN [29] unsupervised image-to-image translation framework to accomplish input-level domain adaptation (Figure 7.9).

The framework proposed by Zhu *et al.* [29] is built on top of a pair of generative adversarial models, concurrently performing conditional image translation between a couple of domain sets, in both the source-to-target and target-to-source directions. The two adversarial modules are further tied by a cycle-consistency constraint, which encourages the cross-domain projections to

be one the inverse of the other. This reconstruction requirement is essential to preserve structural geometrical properties of the input scene, but provides no guarantees about the semantic consistency of translations. In fact, while retaining geometrical coherence, the mapping functions could completely disrupt the semantic classification of input data.

Taking this into account, a number of works [30, 274–276, 298] address semantic consistency by taking advantage of the semantic discriminative capability of the segmentation network. In particular, cross-domain image translations are forced to preserve semantic content as perceived by the semantic predictor, which represents a measure of semantic discrepancy between an original image and its translated counterpart, which is minimized in the optimization of the translation network. Still, with the prediction maps being intrinsically flawed, especially in the target domain where annotations are missing, the inaccurate semantic information provided to the generative module could hurt the learning of the image projections. Thus, some works propose to simultaneously optimize the generative and discriminative framework components in a single stage [30], or even split the segmentation network into separate source and target predictors [276]. Li *et al.* [274] further extend the CycleGAN-based adaptation strategy formulating a bidirectional learning framework. The image-to-image translation and segmentation modules are alternately trained, in an optimization scheme by which each module is provided with positive feedback from the other. The segmentation network benefits from the target-like translated source images with original supervision, while the generative network is aided by the predictor in retaining semantic consistency. This closed-loop structure effectively allows for a progressive adaptation, with both image-to-image translations quality and semantic prediction accuracy gradually enhanced.

Other works [300, 301] resort to different approaches to provide semantic-awareness to the CycleGAN-based adaptation. Li *et al.* [300] propose to assist the cycle-consistent image-to-image translation framework by a soft gradient-sensitive loss to preserve semantic content in the cross-domain projection focusing on semantic boundaries. The idea behind this approach is that, no matter how low-level visual attributes change between domains, the edges defining semantic uniform regions should be easily detectable, regardless of the distribution the image is drawn from. Thus, a gradient-based edge detector should discover consistent edge maps between original images and their transformed versions. In addition, following the intuition that semantically different regions of an image should face a different adaptation, they devise a semantic-aware discriminator structure. In doing so, the discriminator can semantically-wise evaluate resemblance between original and translated samples.

Very recently, Yang *et al.* [301] introduced a phase consistency constraint to the CycleGAN pixel-level adaptation module, observing that the semantic content of an image is mostly encoded in the phase of its Fourier transform, whereas alterations of the amplitude to the representation in frequency does not change its composition.

With a different adaptation perspective, Gong *et al.* [302] adapted the CycleGAN model to generate a continuous flow of domains ranging from source to target ones, by conditioning the generative networks with a continuous variable representing the domain. The reason behind the retrieval of intermediate domains spanning between the two original ones is to ease the adaptation task, by progressively characterizing the domain shift affecting the input data distributions. Moreover, they suggest that resorting to target-like training data from diverse target-like domain distributions improves the generalization capability of the segmentation network.

To reduce the computational burden of the bi-directional structure of CycleGAN (which entails a total of at least four neural networks to be added to the semantic predictor) other works [283, 304–306] discard the backward source-to-target projection branch, seeking for a more light-weight input-level adaptation module, still based on generative adversarial framework. The translation consistency is granted, for example, by the correlation to a related task (*e.g.*, depth es-

timation) [283,304], which is jointly addressed with the semantic segmentation. Choi *et al.* [305], instead, improve the generator of the original GAN framework with feature normalization modules at multiple depths to provide style information to source representations, whereas source content is preserved. Furthermore, a semantic consistency loss from a pre-trained segmentation network promotes coherence of image translations, providing, in fact, a regularizing effect in absence of the cycle-consistency one. Hong *et al.* [306] use a conditional generative function to model the residual representation between source and target feature maps, which is optimized in an adversarial framework. In doing so, they avoid any reliance on a shared domain-invariant latent space assumption, which may not be satisfied due to the highly structured nature of semantic segmentation. The generator takes as input low-level source feature maps, together with a noise sample, and is encouraged to produce high-level feature maps with target-like distribution by a discriminator, which expresses a measure of statistical distance between original and reproduced target representations. Both source original and domain-transformed representations are provided to a dense classifier to compute the cross-entropy loss.

In order to lessen the bias towards the source domain, Yang *et al.* [286] resort to the target-to-source image-to-image translation, in place of the more common source-to-target one, generally employed to generate a form or target supervision from source translated data. The source-like target images are then employed in the supervised training of the predictor thanks to pseudo-labeling. In addition, training the segmentation network directly in the source domain allows to fully exploit the original source annotations, avoiding the risk of semantic alterations, which may happen in the source-to-target pixel-level adaptation scenario. Moreover, to align feature representations between domains, they introduce a label-driven reconstruction network. However, differently from the feature-based reconstruction techniques [279–281] (Section 7.3.2), the generative recreation of input images is performed starting from semantic maps from the segmentation output. In doing so, they seek to guide a category-wise alignment of the segmentation network embeddings, since reconstructions that deviate from their target are penalized, thus providing semantic consistency to network predictions.

A different category of adaptation strategies explores style-transfer techniques to achieve image-level appearance invariance between source and target domains. These approaches are based on the principle that every image can be disentangled into two separate representations, namely content and style. As the style encodes low-level domain-specific texture information, the content expresses domain-invariant high-level structural properties. Thus, being able to combine style properties from target data with semantically preserving source content should effectively allow for the construction of target-distributed training data, still retaining original source annotations. Some techniques [284, 307] involve content and style decomposition in the latent space. Translating a source image, then, means extracting its feature content representation and recombining it with a random target style representation. In a recent work [307], the authors perform multi-modal source-to-target image translation based on the MUNIT architecture [308]. The original datasets are augmented with additional web-crawled data, in order to reduce the gap in terms of task-unrelated data properties between sets, while at the same time highlighting the relevant task-related visual features to be matched. Furthermore, the style transfer method allows for multi-modal translation, *i.e.*, multiple target styles can be transferred to a single source image, thus increasing training data diversity and, in turn, enforcing the adaptation robustness.

Other works [288,309–312] completely avoid the computational complexity of generating high resolution images with GANs by exploiting different types of style transfer techniques. Zhang *et al.* [288] adopt traditional techniques of neural style transfer [313,314] to separate style (low-level feature) from image content (high-level features). In particular, multi-level response maps of a pre-trained CNN are exploited for image synthesis, where image style is expressed by the cor-

relation between feature maps in the form of Gram matrices. Alternative approaches [305, 310] opt for the re-normalization of source feature maps, so that their first and second order statistics match those of the target ones, by means of the AdaIN module [315]. Differently, Dundar *et al.* [311] make use of a photo-realistic style transfer algorithm for an iterative optimization by which both the segmentation network and the translation algorithm performance is constantly improved. Yang *et al.* [312] remove domain-dependent visual attributes from source images by replacing the low-level frequency spectrum with that of target images, without affecting high-level semantic interpretability. They argue that this simple approach, despite not requiring any additional learnable module, results in a remarkably robust adaptation performance when embedded in a multi-band framework that averages predictions with different degrees of spectral alteration. Finally, some approaches [316,317] employ cyclic consisten GANs to also enforce the physics of parts of the scenes.

7.3.4 Classifier Discrepancy

As discussed in Section 7.3.2, feature-level adversarial domain adaptation in its original form entails the competition between the task feature extractor and a domain critic (the discriminator), whose supervisory action in principle should guide towards the cross-domain alignment of feature representations. Task-discrimination instead is granted by a source supervised task objective (*i.e.*, the standard cross-entropy loss for semantic segmentation).

As highlighted by [318,319], the major drawback of this primary form of adversarial adaptation lies in the lack of semantic awareness from the domain discriminator network. Even when the critic manages to grasp a clear expression of marginal distributions, thus effectively leading to a global statistical alignment, category-level joint-distributions necessarily remain unknown to the domain discriminator, as it is not provided with semantic labels when discriminating feature representations. A side effect of this semantic-unaware adaptation is that features can be placed close to class boundaries, increasing the chances of incorrect classification. Furthermore, target representations may be incorrectly transferred to a semantic category different from the actual one in the domain invariant adapted space (negative transfer), as decision boundaries are ignored in the adaptation process.

To overcome these issues, Saito *et al.* [318] propose an Adversarial Dropout Regularization (ADR) approach for UDA to provide cross-domain feature alignment away from decision boundaries. To do so, they completely revisit the original domain adversarial scheme, by providing the task-specific dense classifier (*i.e.*, the encoder) with a discriminative role. In particular, by means of dropout, the classifier is perturbed in order to get two distinct predictions over the same encoder output. Since the prediction variability is subject to an inverse relationship with the proximity to decision boundaries, the feature extractor is forced to produce representations far from those boundaries by minimizing the discrepancy of the two output probability maps. At the same time, the classifier has to maximize its output variation, in order to boost its capability to detect less-adapted features. In this redesigned adversarial scheme, the dense classifier is trained to be sensitive to semantic variations of target features, as to capture all the information stored in its neurons, which in turn are encouraged to be as diverse as possible from each other by the adversarial dropout maximization. On the other hand, the encoder is focused on providing categorical certainty to extracted target features, since removing task-unrelated cues weakens the possibility to achieve dissimilar predictions from the same latent representations.

Following the same principle of Adversarial Dropout Regularization, other approaches resort to adaptation techniques based on classifier discrepancy to achieve a semantically consistent alignment [285, 299, 319–321]. Saito *et al.* [319] improve the framework in [318] by modifying the way of accessing multiple predictions over the same latent space. In place of dropout on classifier's

weights, they introduce a couple of separate decoders, which are simultaneously trained with source supervision, while being forced to produce dissimilar predictions by the maximization of a discrepancy loss. The objective is to avoid the noise sensitivity acquired by the single classifier in ADR, which is essential for the individual decoder to capture the proximity to the support of target samples, but requires an additional training stage to correctly learn the segmentation model as a whole.

The co-training strategy of exploiting a couple of distinct classifiers to infer the degree of target adaptation is further merged to the more traditional generator-discriminator adversarial framework by Luo *et al.* [285]. They use the discrepancy map from the two classifiers' output to weight the adversarial objective. Thereby semantic inconsistent regions highlighted by strong prediction variability get a major focus in the objective, as they should suffer from a more prominent domain shift. Additionally, Lee *et al.* [321] re-propose a form of adversarial dropout to get divergent predictions from a single classifier. However, they drop the adversarial scheme for a non-stochastic virtual dropout mechanism, to discover minimum distance adversarial dropout masks that maximize prediction discrepancy. In the end, they resort to a single unified objective, for a combined optimization of the encoder and decoder to align features between domains, while progressively pushing dense regions and decision boundaries far away from each other.

7.3.5 Self-Training

The self-training strategy entails using highly confident network predictions inferred on unlabeled data to generate pseudo-labels, to be used, in turn, to reinforce the training of the predictor with the self-taught supervision. This approach has been commonly employed in semi-supervised learning (SSL) [322] to exploit additional unlabeled data in order to improve the prediction accuracy. Recently, self-training techniques have been extended to address unsupervised domain adaptation, since UDA can be considered as a variant of the SSL task, even though the additional complexity of UDA from the statistical shift of the unlabeled target data must be further taken into account. Indeed, concurrently learning from source annotations and target pseudo-labels implicitly promotes feature-level cross-domain alignment, while still retaining the task specificity. On the contrary, lacking a unified loss, other adaptation approaches, as the most successful adversarial ones, have to take care of the task-relatedness with additional training objectives. The critical point is that this strategy is self-referential, so careful arrangements must be adopted to avoid catastrophic error propagation. Self-training, in fact, naturally promotes more confident predictions, as the network probability output is encouraged to reach a peaked distribution (at the limit a Dirac distribution) close to the one-hot pseudo-labels. Since no form of external supervision is available on unlabeled target data, the network could yield over-confident predictions by wrongly classifying uncertain pixels. In turn, the iterative self-teaching strategy enforces prediction mistakes, through a propagation mechanism that makes the output progressively deviating from the correct solution. For this reason, the majority of self-training based adaptation approaches rely on various forms of pseudo-label filtering, to allow self-learning only from top confident target predictions, which are implicitly assumed to have a higher chance of being correct.

A first class of adaptation solutions based on self-training [274, 323, 324] employs offline techniques for pseudo-label computation: at every update step a confidence threshold is computed by looking at the entire training set. Target segmentation maps are then directly filtered according to some confidence-based thresholding policy and used in combination with original source annotated data for the supervised learning of the segmentation network.

In this regard, Zou *et al.* [323] propose one of the first UDA techniques based on self-training. They devise an iterative self-training optimization scheme, which alternates steps of segmentation

network training on both source original and target artificial supervision and target pseudo-label estimation. In particular, the target pseudo-labels are treated as discrete latent variables to be computed through the minimization of a unified training objective. In addition, motivated by the fact that class-unaware pseudo-labels confidence filtering is intrinsically biased towards the easy (*i.e.*, more confident) classes, they devise a class-balancing strategy by setting category-wise confidence thresholds. This should promote inter-class balance, as the same amount of top confident pixels are considered for each class, thus resulting in class-wise uniform contributions to the learning process. Finally, since source and target domains are supposed to share high-level scene layout, they also utilize spatial priors from source label statistics, which are inferred for each semantic category and incorporated in the training objective. More recently, Zou *et al.* [324] revisited their previous work in [323] by extending the pseudo-label space from one-hot maps to a continuous space defined by a probability simplex. In this way, by avoiding clear-cut overconfident self-supervision in the whole input image, the effect of the inherent misleading incorrect pixel predictions should be effectively reduced. A continuous pseudo-label space further allows them to introduce a confidence regularizing term in the training objective targeting both pseudo-label (treated as latent variables) and network weights, with the purpose of achieving output smoothness in place of sparse segmentation maps.

In order to avoid slow offline dataset-wise processing, Pizzati *et al.* [307] introduce self-training with weighted pseudo-labels. A learnable confidence threshold is employed for both pseudo-label refinement and weighting, thus making pseudo-labels belong to a continuous space, while concurrently balancing the impact of uncertain pixels. Target weighted self-generated labels are computed over a single batch, but still retain a global view, since the confidence threshold is learned throughout the entire training phase.

A different group of research [26–28] construct a self-training strategy on top of an adversarial discriminative adaptation module applied over the segmentation network output. In particular, on the belief that the fully convolutional discriminator can be regarded as performing a measure of reliability of network estimations, they exploit the discriminator output to identify reliable target predictions, which are then preserved in the pseudo-label filtering operation. In [27] the pseudo-label selection mechanisms is further improve by a region growing strategy. Moreover, Spadotto *et al.* [28] propose to adopt a class-wise adaptive thresholding approach. They select the same fraction of highly confident target pixels for each semantic class, by looking at the batch-wise distribution of the discriminator probability output. In doing so, they provide the adaptation framework with both inter-class confidence flexibility and time adaptability over the training phase.

Another line of works [298, 305, 312, 325] utilize various forms of prediction ensembling to yield more reliable predictions over target data, on top of which pseudo-labeling is performed. Chen *et al.* [325] enhance the adaptation of low-level features by introducing an additional ASPP dense classification module. Hence, self-produced guidance in the form of pseudo-labels from the combined knowledge of low and high level target predictions is exploited as an additional training objective. Yang *et al.* [312] train multiple instances of the segmentation network with multi-band spectrum adaptation to obtain distinct semantic predictors. Then, target pseudo-labels are generated from the mean prediction of the different segmenter instances, resulting in a more robust adaptation when dealing with multiple rounds of self-training.

Rather than operating directly on the predictor output, other self-training approaches [298, 305] resort to an additional network to produce self-guidance over the unlabeled samples. Choi *et al.* [305] propose a self-ensembling adaptation technique, by which a teacher network derived from student network's weights average yields predictions the student network is compelled to follow. In other words, an auxiliary predictor (the teacher network) is providing a sort of pseudo-labels, which are then used to transfer reliable knowledge to the actual predictor (the student network)

by supervised training on target data. With regularization purposes, Gaussian noise is additionally injected on input target images and dropout weight perturbation is applied to the segmentation network to improve adaptation robustness, as student-teacher prediction consistency is enforced even under different random disturbance. Recently, the student-teacher self-ensembling adaptation approach is extended by Zhou *et al.* [298], with the introduction of an uncertainty module that filters out unreliable teacher predictions by looking at self-information maps.

7.3.6 Entropy Minimization

As already pointed out, semi-supervised learning and unsupervised domain adaptation are closely related tasks: indeed, once source and target distributions are matched, the UDA task merely scales down to learning from an unlabeled subset of the training data. Therefore, it is natural that SSL approaches may inspire domain adaptation strategies, as discussed for self-training (Section 7.3.5). Among the successful techniques used to address semi-supervised learning, entropy minimization has been recently introduced to UDA [151]. The principle behind minimizing target entropy to perform domain adaptation follows the observation that source predictions are likely to show more confidence, which in turn translates into high entropy probability outputs. On the contrary, the segmentation network is likely to display a more uncertain behavior on target-distributed samples, as target prediction entropy maps happen to be overall quite unstable, typically being the noise pattern not confined just to the semantic boundaries. Thus, forcing the segmentation network to mimic the over-confident source behavior when applied to the target domain too, should effectively reduce the accuracy gap between domains. In other words, entropy minimization aims at penalizing classification boundaries in the latent space crossing high density regions, while jointly encouraging well-clustered target representations properly sorted out by decision boundaries.

In its simplest fashion [151] entropy minimization is performed at a pixel-level, so that each spatial unit of the prediction map brings an independent contribution to the final objective. However, the basic approach suffers from some intrinsic limitations, demanding further arrangements to boost the adaptation performance [151,312,325]. To leverage structural information of semantic maps, Vu *et al.* [151] propose a global adversarial optimization to enforce distribution alignment over source and target entropy maps. In doing so, they rely on a domain discriminator to capture global patterns differentiating samples from separate domains, thus achieving a more semantically meaningful cross-domain match of entropy behavior. Class-wise priors on label distributions inferred from source annotations are further enforced on target predictions to avoid class imbalance towards easy classes.

In a following publication, Chen *et al.* [325] observe that the entropy minimization objective can be seriously hindered by the gradient predominance of more confident predictions. Indeed, moving from high to low uncertainty areas, the gradient rapidly increases, and its value tends to infinity as the output probability distribution tends to the delta function. This probability imbalance in general prevents the segmentation network from learning over areas with little accuracy, in which the gradients result much lower than those of easy-to-transfer image regions. To address this issue, they devise a maximum squares loss, which produces a gradient signal that grows linearly with the input probability. They also face class unbalance by introducing a category-wise weighting factor based on target distribution from prediction maps in place of source annotations, as they argue that source class statistics may significantly deviate from target ones.

Very recently, Yang *et al.* [312] added an entropy minimization technique as an additional module to their adaptation scheme. The intent is to seek a regularization effect over the training on unlabeled target data, accomplished by pushing the decision boundaries away from high-

density regions in the target latent space, with basically no overhead to the actual framework. The strength of the approach is enhanced by the combined application of other adaptation modules to achieve domain alignment. This, in fact, shifts the UDA task towards SSL, thus making entropy minimization more effective. Moreover, to avoid excessive emphasis on low entropy predictions, they adopt a penalty function that increases the focus on less-adapted high entropy regions of target images.

Entropy minimization has been used together with feature space shaping techniques in a few recent works [31–33]. In [31], besides using entropy minimization, internal feature representations are forced to be clustered, sparse and orthogonal (if belonging to different classes) in both source and target domains to improve feature-level adaptation. In other recent works [32, 33] a norm alignment constraint is further introduced to aid a class-wise feature orthogonality objective in promoting disjoint sets of active feature channels between distinct semantic categories, while driving target embeddings towards the highly confident (*i.e.*, associated with high values of feature norm) source distribution.

7.3.7 Curriculum Learning

Another research area regards curriculum learning approaches, where some easy tasks are solved first, inferring some important and useful properties related to the target domain. Then, this information is used to support the training of a network dealing with a more challenging task, like image segmentation. This family of approaches shares many similarities in spirit with self-training. The main difference between the two approaches lies in the content of the pseudo-labels. While in the self-training approaches the pseudo-label is an estimate of the desired annotation on the target set and it is used as such during training, in curriculum approaches the pseudo-label is represented by some inferred statistical properties of the target domain (different from the labels for the task) and the network is trained to reproduce such inferred properties in the target predictions.

The first work of this family is [326] and its extension [327], where a couple of easy tasks that are less sensitive to domain discrepancy are solved: namely, the label distribution over global images and the label distribution over local landmark superpixels. The former property is evaluated in the source domain, as the number of pixels in the labels associated to each category, normalized by the total number of pixels. On the other hand, target labels are not available in unsupervised domain adaptation and consequently a machine learning model should be trained on the source domain to estimate them. In the papers, it is argued that this task can be solved more easily than image segmentation and that the results can be used to guide the adaptation of the segmentation task. To estimate the first property on the target domain, a logistic regression model is employed. While the first property is useful to guarantee that the ratio among different categories matches the ones of the target domain, samples with semantic maps not following the estimated label distribution on the target domain are still penalized. To solve this problem, a second clue is introduced. Images are divided into superpixels and an SVM classifier is used to select the most representative anchor superpixels and the label distribution is estimated over them. The final objective is a mixture of the pixel-wise cross-entropy of the source samples and the cross-entropy on the two properties on the target domain discussed before. In [328, 329], a technique to adapt the domain of a segmentation model from clear weather to dense fog images is introduced. A novel method, called Curriculum Model Adaptation (CMAda), is proposed to gradually adapt the model to segment images with an incrementally growing amount of fog. A new method to add synthetic fog to images and a new fog density estimator are introduced. It is important to remark that the fog generator has a tunable parameter β that controls the density of the fog to add to the images. This made it possible to generate samples from the dataset

Cityscapes with different synthetic fog density and to use them to train an AlexNet model [330] to perform a regression problem to discover β from images. The trained fog density estimator, then, can also be used to estimate the fog density of real foggy images. The algorithm presented starts from a source domain of clear weather images and progresses through intermediate target domains of incrementally denser fog, and, finally, reaches the target domain of dense fog images. During training, the labels of source domain and of synthetic fog images are available, while the real foggy images are unlabeled. The segmentation model, initially, is pre-trained with supervision on the source domain and then, with as many adaptation training steps as the number of denser fog steps, it is gradually shifted towards the target domain. Starting from the assumption that images with lighter fog are easier to segment, the model of the current step is used to evaluate the labels for real foggy images with intensity less than the *beta* of the current step. Then these samples are used together with images with synthetic fog of density *beta* of the next step to train the model with supervision. Iterating this process for all the steps towards the target dataset, the model adapts, in an unsupervised way (labels of the real foggy images are not used), to segment real dense fog images. In [331], the connection between curriculum learning and self-training is highlighted and a method (called self-motivated pyramid curriculum domain adaptation, PyCDA) that uses and merges both techniques is presented. The authors remind that in self-training there are two main training steps that alternate: (1) the evaluation of pseudo-labels for the target domain and (2) the supervised training of the segmentation network with the labeled source domain images and with the target domain images with pseudo-labels. In curriculum learning there are also two steps that alternate: (1) the inferring of properties of the target domain (*e.g.*, frequency label distributions over global images or image regions, like superpixels) and (2) the update of the network parameters using the labeled source domain and the target domain inferred properties. In PyCDA the two approaches are merged: the pseudo-labels used in self-training are considered as a property of the curriculum approach. The papers also substitute the superpixels used in [326] with small squared regions to improve the algorithm efficiency and also all the curriculum properties are inferred with the segmentation networks themselves and additional models (for example, SVMs or logistic regression models) are not needed.

7.3.8 Multi-Tasking

Some works exploit additional types of information available in the source domain dataset, for example, depth maps, to improve the performance in the target domain. In other words, the models are trained to solve additional tasks (for example depth regression) simultaneously to image segmentation in order to build an invariant and generic embedding of the images. In [304], the authors highlight that when the source domain is made of synthetic data we could include other information about the dataset samples beyond the semantic maps, for example, depth maps. This is called Privileged Information (PI) and it includes all properties that may be useful for the training. The method proposed in [304] is called Simulator Privileged Information and Generative Adversarial Networks (SPIGAN), which uses an adversarial learning scheme performing source-to-target image translation together with a network trained on source images and on adapted images that tries to predict their privileged information (*e.g.*, the depth map). In particular, the PI is used as regularization for the domain adaptation. A different use of the extra depth information in the source domain to enhance the appearance features and improve the alignment of the source-target domains is presented in [291]. The method introduced is called Depth-Aware Domain Adaptation (DADA) and includes a specific architecture and a learning strategy. The architecture starts from an existing segmentation network and includes some extra modules to predict the monocular depth and to feed the information of this task back

in the main stream. Residual auxiliary blocks are used for this purpose. To perform domain adaptation, images from source and target domains are fed to the network to compute the class-probability and depth maps. Then, the former is processed into self-information maps and merged to the latter to produce depth aware maps. Finally, these maps are used in an adversarial training to adapt the source domain. It is important to remark that the depth information is not used as regularization, but it is directly considered while deriving prediction for the main task. In the paper, it is argued that this is a more explicit and more useful way to exploit the depth information than the method presented in [304].

A third different use of the depth maps is introduced in [283] where a method called Geometrically Guided Input-Output Adaptation (GIO-Ada) is presented. The geometric information is exploited to improve the adaptation both at the input-level and at the output-level. The former adaptation tries to reduce the visual differences of the images of the source and target domains. A transform network accepts as input source images together with their semantic maps and depth maps to compute adapted images, visually similar to images of the target domain. A discriminator is used in an adversarial learning with the transform network to distinguish real target domain images from adapted ones. The main contribution of the paper in this adaptation is the use of semantic maps and depth maps as additional inputs for the transform network. The output-level adaptation is built with a task network that computes, for each input, the semantic map and the depth map. Such outputs are fed to an additional discriminator which tries to distinguish whether they were computed from a real or adapted image. In [320], a network composed of a feature generator followed by two classifiers (that computes semantic maps) has been adopted and a maximum classifier discrepancy approach is used for the unsupervised adaptation from a synthetic source domain to a real target domain. Two techniques are presented to improve the performance of the network: a data-fusion approach and a multi-task one. The former merges the RGB image information and the depth information and use the result as an input of the network. In the latter, only RGB images are used as inputs, however three tasks are solved simultaneously by different networks after the feature generator to boost the overall performance of the network in the target domain: namely, semantic segmentation, depth regression and boundary detection.

7.3.9 Latent-Level Regularization

Latent space regularization has been shown to ease the semantic segmentation tasks in different settings, such as UDA [138, 332], continual learning [21] and few-shot learning [140, 141]. The idea is to embed additional constraints on feature representations during the training process, enforcing a regular semantic structure on latent spaces of the deep neural classifier. In UDA, where target semantic supervision is missing, regularization can be applied in class-conditional manner by relying on the exclusive supervision of source samples, while indirectly propagating its effect to target representations as well.

Clustering in UDA. A few approaches have been proposed recently to address UDA in image classification by resorting to a discriminative clustering algorithm, whose goal is to disclose target class-conditional modes in the feature space. A group of works [138, 139, 332, 333] embed variations of the K-means algorithm in the overall adaptation framework, where clusters of geometrically close latent representations are identified, revealing the semantic modes from the unlabeled target domain. Being developed to address image classification, these clustering strategies may lose efficacy when dealing with semantic segmentation. Moreover, they deeply rely on a geometric measure of feature similarity to assign target pseudo-labels, which may not be feasible in a very high-dimensional space. For this reason, this type of clustering technique is often combined with a learnable projection to discover a more easily tractable lower dimensional

latent space [139, 332, 333].
To overcome the lack of target semantic supervision, other approaches [145,146] resort to pseudo-labels directly from network predictions to discover target class-conditional structures in the feature space. Those structures are then exploited to perform a within-domain feature clusterization [146] and cross-domain feature alignment by centroid matching [145, 146]. Starting from analogous premises, [32, 33] extend a similar form of inter and intra class adaptation to the semantic segmentation scenario, by introducing additional modules that help to address the inherent increased complexity.

Avoiding the need for target pseudo-labels, [244, 250] propose a self-supervised clustering technique to discover target modes without any form of supervision. However, their approach is not easily scalable to semantic segmentation, as it requires to store feature embeddings of past samples, which is rather impractical when each data instance is associated with thousands of latent representations.
Quite recently, Tang *et al.* [334] argue that a direct class-wise alignment over source and target features could harm the discriminative structure of target data. Thus, they perform intrinsic feature alignment by a joint, yet distinct, model training with both source and target data. Nonetheless, in a more complex semantic segmentation scenario, an implicit adaptation could be not enough to bridge the domain gap that affects the effectiveness of target predictions.
Orthogonality and Sparsity. Deep neural networks are trained to learn a compact representation of the scene. However, no constraint is posed on the orthogonality among feature vectors belonging to different classes or on the sparsity of their activations [335, 336]. The orthogonality between feature vectors has been recently proposed for UDA in [143, 144]. In these works, a penalty term is introduced to force the prototypes (*i.e.*, the class centers) to be orthogonal. Feature-level orthogonality has been also explored in [337] to limit the redundancy of the information encoded in feature representations. Recent works [31–33] promote disjoint sets of active feature channels between distinct semantic categories in order to reduce cross-talk and overlapping channels among features of different classes.

In [31] the idea of reducing interference among features is also translated into a constraint to make features channel-wise sparse. There are not many prior works employing channel-wise sparsification in deep learning models. However, some prior techniques exist for domain adaptation on linear models exploiting sparse codes on a shared dictionary between the domains [147, 148]. Additionally, in [338] an expectation maximization approach is proposed to compute sparse code vectors which minimize the energy of the model. Although the approach is applied to linear encoders and decoders for simplicity, it could be extended to non-linear models.

7.3.10 New Research Directions

Unsupervised Domain Adaptation in its original interpretation aims at addressing the domain shift by transferring representations concerning a specific and well-defined set of semantic categories shared across source and target data. This follows the assumption that the target domain contains only instances of the classes which can be found in source samples. Nevertheless, while being a reasonable hypothesis that does not hinder the generality of the adaptation task, in practice it is common that images from a novel domain may contain objects from unseen categories.

Moving in the direction of a more general definition of the adaptation objective, some works have tackled the open-set domain adaptation [238] applied to the image classification task, which entails unknown categories peculiar to the target domain not present in the source one, but they still retain a somewhat strict prior definition of the adaptation settings class-wise. Recently, a few novel approaches [239, 244] have proposed to relax the common premises on the domain

adaptation settings, to effectively move towards a more realistic scenario where little can be inferred a priori about target data properties, thus widening the applicability to real-world solutions.

Saito *et al.* [244], for instance, introduce the Universal Domain Adaptation problem, allowing for basically no beforehand characterization of target classes. In particular, they resort to a neighborhood clustering technique to assign each target sample to either a source class or to the unknown category without any supervision. Then, the matching of cross-domain representations is enforced by entropy minimization to achieve domain alignment.

A step further is attained by solving the recognition of unseen target categories, which have to be individually learned rather than simply acknowledged as unknown. Zhuo *et al.* [239] address what they call the Unsupervised Open Domain Recognition task, where the objective is to learn to correctly classify target samples of unknown classes. To do so, they reduce the domain shift between source and target sets by an instance matching discrepancy minimization, weighted according to feature similarity. Once the semantic predictor has achieved domain invariance, classification knowledge can be safely transferred from known to unknown categories by a graph CNN module.

Despite that the aforementioned adaptation approaches have proven to be quite effective for the image classification task, further adjustments need to be done to deal with the additional complexity of feature representations of a semantic segmentation network. In this regard, a novel Boundless Unsupervised Domain Adaptation (BUDA) task is proposed by Bucher *et al.* [245] specifically for semantic segmentation. Similarly to UODR [239], the standard domain adaptation problem is *unbounded* to explicitly handle instances of new unseen target classes, while relying solely on a minimal semantic prior in the form of class names, which are supposed to be known in advance. Thus, the overall task is decoupled in the domain adaptation and zero-shot learning problems. First, the domain adaptation of categories in common between source and target domains is performed, via an entropy minimization technique carefully designed to avoid incorrect alignment over unseen target classes. Then, a zero-shot learning strategy [339] is exploited to transfer knowledge from seen to unseen classes, by a generative model able to synthesize visual features conditioned by class descriptors.

Another closely related research direction, which is currently gaining wider and wider interest among the research community, is the continual learning task. As we have seen in previous chapters, continual learning could be regarded as a particular case of transfer learning, where the data domain distribution changes at every incremental step and the models should perform well on all the domain distributions. For instance, in class-incremental learning, the learned model is updated to perform a new task whilst preserving previous capability. Initially proposed for image classification [49, 79] and object detection [51], it has been recently explored also for semantic segmentation [18, 20, 23]. Another formulation of this problem regards the coarse-to-fine refinement of semantic labels, in which previous knowledge acquired on a coarser task is exploited to perform a finer task, hence modifying the labels distribution [24, 25].

7.4 A Case Study: Synthetic to Real Adaptation for Semantic Understanding of Road Scenes

The main aspect for UDA techniques is the ability to transfer knowledge acquired on one dataset to a different context. Hence, the considered data play a fundamental role in the design and evaluation of UDA algorithms. In this section, we focus on one of the most interesting application scenarios: *i.e.*, the ability to transfer knowledge acquired on synthetic datasets (source domain), where labels are relatively inexpensive and can be easily produced with computer

graphics engines, to real-world ones (target domain), where annotations are highly expensive, time-consuming and error-prone. Many of the works dealing with this task focus on urban scenes mainly for four reasons:

1. autonomous driving is nowadays one of the biggest research areas and massive fundings support this research [340];

2. many synthetic and real-world datasets are publicly available for this scenario [243, 341–344];

3. autonomous vehicles should fully understand the surrounding environment to plan decisions [345] and such navigation task in the environment could be encountered in many other applications, for example, in the robotics field;

4. the first works on the topic addressed this setting and it has become the de-facto standard for performance comparison with the state-of-the-art in the UDA for semantic segmentation field.

In the autonomous driving scenario, the pixel-level annotation must be manually provided for a huge amount of frames acquired by cameras mounted on cars driving around. This annotation is expensive and requires a huge amount of work. Some recent papers [243, 341] introduced a workaround for this issue using computer generated data for training the networks. The realistic rendering models developed by the video game industry can be used to produce a large amount of high quality rendered road scenes [243].

Before presenting the more commonly used synthetic and real-world datasets, we stress that in the unsupervised domain adaptation scenario the expensive labels of real samples in the target domain are not needed for training. However, a limited number of real target samples must be manually labeled for testing (and sometimes validating) the performance of the algorithms. On the other hand, large synthetic datasets corresponding to the source domain are equipped with annotations that are exploited for supervised training.

7.4.1 Source Domain: Synthetic Datasets of Urban Scenes

One of the first large scale synthetic datasets for urban driving is the **GTA5** dataset [243]. It contains 24,966 synthetic 1914×1052 px images with pixel-level semantic annotation. The images have been rendered using the open-world video game *Grand Theft Auto V* and are all from the car perspective in the streets of American-style virtual cities (resembling the ones in California). Typically, 23,966 images are used for the supervised training, while 1000 are taken out for validation purposes, as we do in our works. The images have an impressive visual quality and are very realistic since the rendering engine comes from a high budget commercial production. The data is labeled into 19 semantic classes, which are compatible with the ones of real-world datasets as Cityscapes or Mapillary (after a proper re-mapping of labels).

The SYNTHIA-RAND-CITYSCAPES dataset has been sampled using the same simulator as the **SYNTHIA** dataset [341] and contains 9,400 synthetic 1280×760 px images with pixel level semantic annotation. The images have been rendered with an ad-hoc graphic engine, allowing to obtain a large variability of photo-realistic street scenes (in this case they come from virtual European-style towns in different environments under various light and weather conditions). On the other hand, the visual quality is lower than the commercial video game GTA5. The semantic labels are compatible with 16 classes of real-world datasets like Cityscapes and Mapillary. For the evaluation, either 13 or 16 classes are taken into consideration. Typically, 9,300 images are used

for the supervised training while 100 are taken out for validation purposes, as we do in our works.

CARLA (CAR Learning to Act) [346] is an open-source simulator for autonomous driving research built over the Unreal Engine 4 rendering software. It has been designed to grant large flexibility and both the physics and the rendering simulations are quite realistic. Two virtual towns have been designed: *Town 1* with 2.9 km of drivable roads and *Town 2* with 1.4 km of drivable roads. Three-dimensional artists first laid roads and sidewalks, then they placed houses, terrain, vegetation, and traffic infrastructure to resemble a realistic environment. Then, dynamic objects, like cars and pedestrians, spawn from specific coordinates. The APIs of CARLA give access to a semantic segmentation camera that can distinguish 13 different classes. This feature makes it possible to sample urban datasets very quickly, easily and with a large control on the variability of samples. Another relevant aspect is that anyone can create their own dataset based on the specific needs customizing the open-source simulator.

7.4.2 Target Domain: Real-World Datasets of Urban Scenes

The **Cityscapes** dataset [342] contains $2,975$ color images of resolution 2048×1024 px captured on the streets of 50 European cities. The images have pixel-level semantic annotation with a total of 34 semantic classes. For the evaluation of UDA approaches, typically the original training set (without the labels) is used for unsupervised adaptation, while the 500 images in the original validation set are used as a test set (since the test set labels have not been made available).

The **Mapillary** dataset [343] contains $25,000$ variable (typically very high) resolution color images taken from different devices in many different locations around the world. The variability in classes, appearance, acquisition settings and geo-localization makes the dataset the most complete and the one of the highest quality in the field. The 152 object-level semantic annotations are often re-conducted to the classes present in the Cityscapes dataset, for example, following the mapping in [347]. The training images (without the labels) are used for unsupervised adaptation and the images in the original validation set are exploited as a test set.

The **Oxford RobotCar Dataset** [348] contains about 1000 km of images recorded driving in the central part of Oxford (UK). The same route (approximately 10 km long) has been repeatedly traversed for almost a year and 20 million images under different weather and light conditions have been collected by the six cameras the car was equipped with. All data are associated also to LiDAR, GPS and INS ground truth.

The **Cross-City** benchmark [277] is sometimes used in a real-to-real setup, where the Cityscapes dataset takes the role of source domain, while the Cross-City dataset [277] takes the role of target. **Google Street View** is used to collect a large number of not annotated street images from *Rome*, *Rio*, *Tokyo* and *Taipei*. These cities have been chosen to ensure enough visual variations and the locations in the cities have been randomly selected. Using the time-machine feature of *Google Street View* it has been possible to capture images of the same street scene at different times in order to extract static objects priors. Such dataset is comprised of $12,800$ $(4 \times 3,200)$ high resolution (2048×1024 px) images.

7.4.3 Methods Comparison

In this section, the main results of the approaches described in the previous chapters are summarized and briefly discussed. For the sake of brevity, only the most widely used datasets are considered in this section: namely, GTA5 and SYNTHIA as source datasets, and Cityscapes as target datasets. Before digging into the description of the results of existing methods, we warn the reader to be aware that different evaluation protocols and experimental setups exist

making the direct comparison of the final accuracy results not always faithful. For instance, differences in input image resolution, batch size, backbone network architecture and other training parameters may alter the comparison.

Table 7.1: Mean IoU (mIoU) for different methods grouped by backbone in the scenario adapting source knowledge from GTA5 to Cityscapes.

Method	**Backbone**	**mIoU**	**Method**	**Backbone**	**mIoU**
Biasetton *et al.* [26]	ResNet-101	30.4	Chen *et al.* [287]	VGG-16	35.9
Chang *et al.* [284]	ResNet-101	45.4	Chen *et al.* [276]	VGG-16	38.1
Chen *et al.* [287]	ResNet-101	39.4	Choi *et al.* [305]	VGG-16	42.5
Chen *et al.* [325]	ResNet-101	46.4	Du *et al.* [278]	VGG-16	37.7
Du *et al.* [278]	ResNet-101	45.4	Hoffman *et al.* [273]	VGG-16	27.1
Gong *et al.* [302]	ResNet-101	42.3	Hoffman *et al.* [275]	VGG-16	35.4
Hoffman *et al.* [275]	ResNet-101	42.7 *	Huang *et al.* [289]	VGG-16	32.6
Li *et al.* [274]	ResNet-101	48.5	Li *et al.* [274]	VGG-16	41.3
Lian *et al.* [331]	ResNet-101	47.4	Lian *et al.* [331]	VGG-16	37.2
Luo *et al.* [290]	ResNet-101	42.6	Luo *et al.* [290]	VGG-16	34.2
Luo *et al.* [285]	ResNet-101	43.2	Luo *et al.* [285]	VGG-16	36.6
Michieli *et al.* [27]	ResNet-101	33.3	Saito *et al.* [319]	VGG-16	28.8
Toldo *et al.* [31]	ResNet-101	45.9	Toldo *et al.* [31]	VGG-16	34.2
Barbato *et al.* [32]	ResNet-101	41.1	Barbato *et al.* [32]	VGG-16	36.2
Spadotto *et al.* [28]	ResNet-101	35.1	Sankaranarayanan *et al.* [279]	VGG-16	37.1
Tsai *et al.* [282]	ResNet-101	42.4	Tsai *et al.* [282]	VGG-16	35.0
Tsai *et al.* [292]	ResNet-101	46.5	Tsai *et al.* [292]	VGG-16	37.5
Vu *et al.* [151]	ResNet-101	45.5	Vu *et al.* [151]	VGG-16	36.1
Wu *et al.* [309]	ResNet-101	38.5	Wu *et al.* [309]	VGG-16	36.2
Yang *et al.* [312]	ResNet-101	50.5	Yang *et al.* [312]	VGG-16	42.2
Zhang *et al.* [288]	ResNet-101	47.8	Zhang *et al.* [326]	VGG-16	28.9
Zou *et al.* [324]	ResNet-101	47.1	Zhang *et al.* [327]	VGG-16	31.4
Murez *et al.* [280]	ResNet-34	31.8	Zhou *et al.* [298]	VGG-16	47.8
Lian *et al.* [331]	ResNet-38	48.0	Zhu *et al.* [281]	VGG-16	38.1 *
Zou *et al.* [323]	ResNet-38	47.0	Zou *et al.* [323]	VGG-16	36.1
Zou *et al.* [324]	ResNet-38	49.8	Hong *et al.* [306]	VGG-19	44.5
Lee *et al.* [321]	ResNet-50	35.8	Chen *et al.* [276]	DRN-26	45.1
Saito *et al.* [318]	ResNet-50	33.3	Dundar *et al.* [311]	DRN-26	38.3
Wu *et al.* [309]	ResNet-50	41.7	Hoffman *et al.* [275]	DRN-26	39.5
Hoffman *et al.* [275]	MobileNet-v2	37.3 *	Huang *et al.* [289]	DRN-26	40.2
Toldo *et al.* [30]	MobileNet-v2	41.1	Liu *et al.* [349]	DRN-26	39.1 *
Zhu *et al.* [29]	MobileNet-v2	29.3 *	Yang *et al.* [301]	DRN-26	42.6
Murez *et al.* [280]	DenseNet	35.7	Zhu *et al.* [29]	DRN-26	39.6 *
Huang *et al.* [289]	ERFNet	31.3	Saito *et al.* [319]	DRN-105	39.7

*: values from results of competing works.

Table 7.1 shows the mIoU results for different methods grouped by the employed backbone network when adapting source knowledge from GTA5 to Cityscapes. In some cases, results of methods of papers that did not directly evaluate on the considered scenario have been extrapolated from comparisons reported in other papers and are marked with an asterisk. We can

appreciate that the results could greatly vary depending on the method and on the evaluation protocol used.

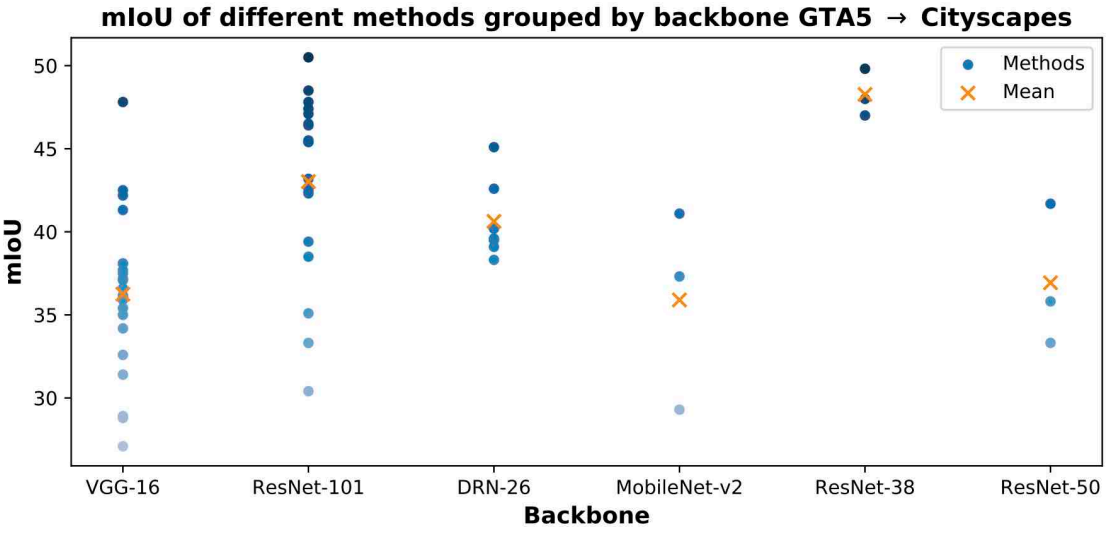


Figure 7.10: Mean IoU (mIoU) of different methods grouped by backbone in the scenario adapting source knowledge from GTA5 to Cityscapes (see Table 7.1). Backbones are sorted by decreasing the number of entries. Orange crosses represent the per-backbone mean mIoU. Only the backbones with 3 or more entries are displayed.

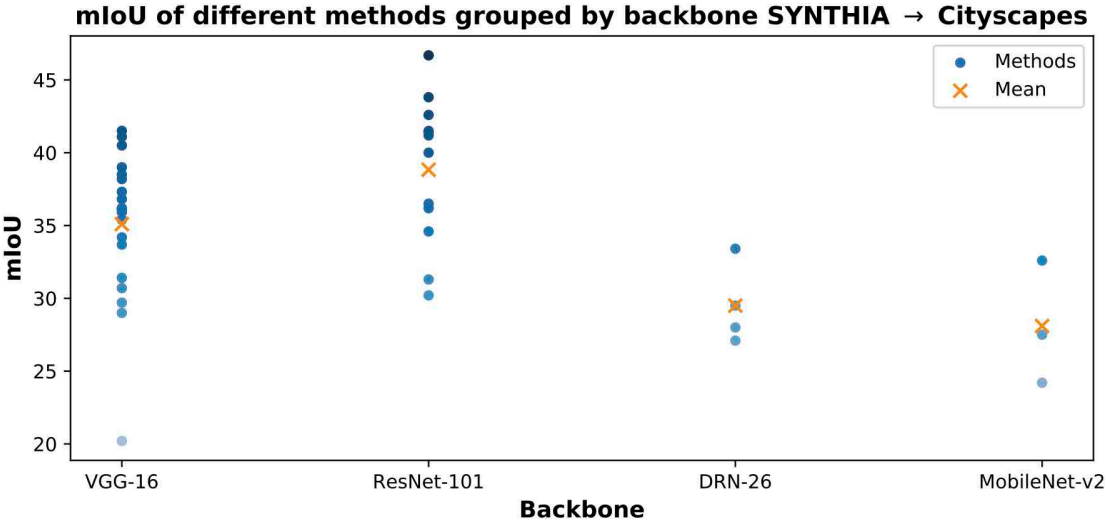


Figure 7.11: Mean IoU on 16 classes (mIoU_{16}) of different methods grouped by backbone in the scenario adapting source knowledge from SYNTHIA to Cityscapes (see Table 7.2). Backbones are sorted by decreasing number of entries. Orange crosses represent the per-backbone mean mIoU. Only the backbones with 3 or more entries are displayed.

The entries of this table are scattered in Figure 7.10 grouped by backbone architecture to show the mIoU values and the corresponding mean (only the backbones with at least three entries are considered in the plot). In general, we can see that ResNet-based approaches outperform the competitors. Moreover, the most widely diffused architectures are ResNet-101 and VGG-16.

Table 7.2: Mean IoU (mIoU) for different methods grouped by backbone in the scenario adapting source knowledge from SYNTHIA to Cityscapes. The table reports the mIoU computed over 13 or 16 semantic classes depending on the label set employed.

Method	Backbone	mIoU_{13}	mIoU_{16}	Method	Backbone	mIoU_{13}	mIoU_{16}
Biasetton *et al.* [26]	ResNet-101	-	30.2	Chen *et al.* [277]	VGG-16	35.7	-
Bucher *et al.* [245]	ResNet-101	-	36.2	Chen *et al.* [287]	VGG-16	-	36.2
Chang *et al.* [284]	ResNet-101	-	41.5	Chen *et al.* [287]	VGG-16	41.8 *	36.2 *
Chen *et al.* [325]	ResNet-101	48.2	41.4	Chen *et al.* [276]	VGG-16	-	38.2
Du *et al.* [278]	ResNet-101	50.0	-	Chen *et al.* [283]	VGG-16	43.0	37.3
Li *et al.* [274]	ResNet-101	51.4	-	Choi *et al.* [305]	VGG-16	46.6	38.5
Lian *et al.* [331]	ResNet-101	53.3	46.7	Du *et al.* [278]	VGG-16	43.4	-
Luo *et al.* [290]	ResNet-101	46.3	-	Hoffman *et al.* [273]	VGG-16	17.0	20.2 *
Luo *et al.* [285]	ResNet-101	47.8	-	Huang *et al.* [289]	VGG-16	-	30.7 *
Michieli *et al.* [27]	ResNet-101	-	31.3	Lee *et al.* [304]	VGG-16	42.4 *	36.8
Toldo *et al.* [31]	ResNet-101	48.2	41.1	Toldo *et al.* [31]	VGG-16	43.7	37.1
Barbato *et al.* [32]	ResNet-101	48.1	41.7	Barbato *et al.* [32]	VGG-16	43.5	39.4
Spadotto *et al.* [28]	ResNet-101	-	34.6	Li *et al.* [274]	VGG-16	-	39.0
Tsai et al [292]	ResNet-101	46.5	40.0	Lian *et al.* [331]	VGG-16	42.6	35.9
Tsai *et al.* [282]	ResNet-101	46.7	-	Luo *et al.* [285]	VGG-16	39.3	-
Vu *et al.* [151]	ResNet-101	48.0	41.2	Luo *et al.* [290]	VGG-16	37.2	-
Vu *et al.* [291]	ResNet-101	49.8	42.6	Sankaran. *et al.* [279]	VGG-16	42.1 *	36.1
Wu *et al.* [309]	ResNet-101	-	36.5	Tsai et al [292]	VGG-16	39.6	33.7
Yang *et al.* [312]	ResNet-101	52.5	-	Tsai *et al.* [282]	VGG-16	37.6	-
Zou *et al.* [324]	ResNet-101	50.1	43.8	Vu *et al.* [151]	VGG-16	36.6	31.4
Zou *et al.* [323]	ResNet-38	-	38.4	Wu *et al.* [309]	VGG-16	-	35.4
Wu *et al.* [309]	ResNet-50	48.4	42.5	Yang *et al.* [312]	VGG-16	-	40.5
Hoffman et al [275]	MobileNet-v2	-	27.5 *	Yang *et al.* [301]	VGG-16	48.7	41.1
Toldo *et al.* [30]	MobileNet-v2	-	32.6	Zhang *et al.* [326]	VGG-16	34.8 *	29.0
Zhu *et al.* [29]	MobileNet-v2	-	24.2 *	Zhang *et al.* [327]	VGG-16	-	29.7
Chen *et al.* [276]	DRN-26	-	33.4	Zhou *et al.* [298]	VGG-16	48.6	41.5
Dundar *et al.* [311]	DRN-26	-	29.5	Zhu *et al.* [281]	VGG-16	40.3 *	34.2 *
Liu *et al.* [349]	DRN-26	-	28.0*	Zou *et al.* [323]	VGG-16	36.1	35.4
Zhu *et al.* [29]	DRN-26	-	27.1 *	Hong *et al.* [306]	VGG-19	-	41.2
Saito *et al.* [319]	DRN-105	43.5 *	37.3 *				

*: values from results of competing works.

Employing the ResNet-101 architecture looks to be the best option for full comparison with the existing literature.

Similarly, Table 7.2 reports the results grouped by backbone when adapting source knowledge from SYNTHIA to Cityscapes. The table reports two setups, *i.e.*, considering either 13 or 16 classes, since both are often considered in this case. The respective entries of mIoU_{16} are scattered in Figure 7.11 to give an overview of the most widely used techniques and of the results achieved. Also in this scenario, VGG-16 is the most popular architecture followed by ResNet-101, which generally shows higher results.

Some recent works also consider the Mapillary dataset adapting from either GTA5 or SYNTHIA as source datasets, as before. In this case, the comparison is quite limited [26–28] and, to the best of our knowledge, the highest performing approach is [28], which achieves a mIoU of 41.9 when adapting from GTA5.

7.5 Summary

In this chapter, we presented a comprehensive overview of the recent advancements in Unsupervised Domain Adaptation for semantic segmentation. This is a very relevant task since deep learning architectures for semantic segmentation require a huge amount of labeled training samples, which in many practical settings are not available due to the complex labeling procedure. For this reason, a wide range of different UDA approaches for this task have been proposed in the recent years.

In order to organize the wide range of existing approaches we started from grouping them at a high-level, based on where the domain adaptation is performed: namely, at input-level (*i.e.*, on the images provided to the network), at feature-level, at output-level or at some ad-hoc network levels. After this macroscopic subdivision, we moved to the actual review of the literature in the field, dividing the existing works into seven (non mutually exclusive) categories: *i.e.*, based on adversarial learning, on generative approaches, on the analysis of the classifier discrepancies, on self-training, on entropy minimization, on curriculum learning and, finally, on multi-task learning. For each category, we presented the most successful approaches and we summarized the main ideas of each contribution.

Then, we considered a case study: the synthetic to real adaptation for semantic understanding of road scenes. Besides being a very relevant task since it is one of the key enabling technologies for autonomous driving, it has also been used for the evaluation of many papers in the field and we concluded comparing the accuracy of many different works grouped by the backbone architecture on this task.

In the next chapters we will present our contributions to the advancement of this field, ranging from output-level methods in Chapter 8 to input- and feature- level adaptation in Chapter 9.

8 Output-Level Domain Adaptation

8.1 Introduction

This chapter explores novel output-level Unsupervised Domain Adaptation (UDA) strategies. In Section 8.2 we present the first line of research, where we investigate adversarial learning and self-teaching [26, 27]. In Section 8.3 we refine previous strategies by means of an adaptive confidence estimation mechanism over the predicted segmentation map [28].

We test our models on the task of domain adaptation for semantic segmentation of urban scenes from synthetic to real-world domains. In particular, we employ the SYNTHIA and GTA5 synthetic datasets to train the supervised component, whereas we resort to real data from Cityscapes and Mapillary datasets for the unsupervised adaptation modules.

8.1.1 Contributions

We proposed two main approaches derived from the same common framework:

1. In the first approach (Section 8.2) we drive the learning and adaptation process by means of three components [26, 27]: a standard supervised learning loss on labeled source data; an adversarial learning module that exploits both labeled source data and unlabeled target data; finally, a self-teaching strategy applied to unlabeled data. The last component exploits a region growing framework guided by the segmentation confidence. Furthermore, we weighted this component on the basis of the class frequencies to enhance the performance on less common classes.

2. In the second approach (Section 8.3) we refine the previous approach. Indeed, we enriched the adversarial module, which is now driven by a couple of fully convolutional discriminators dealing with different domains: the first discriminates between ground truth and generated maps, while the second between segmentation maps coming from synthetic or real-world data. The self-training module exploits the confidence estimated by the discriminators on unlabeled data to select the regions used to reinforce the learning process. Furthermore, the confidence is thresholded with an adaptive mechanism based on the per-class overall confidence.

8.2 UDA with Adversarial Learning and Self-Teaching

As we discussed in Chapter 7, deep neural networks have shown impressive performance on this task. However, they have the key drawback that a huge amount of labeled data is required for their training, especially in case recent highly complex architectures are used.

Despite the impressive realism of recent video games graphics, there is still a large domain shift between the computer generated data and real-world images acquired by video cameras on cars. To be able to really exploit computer generated data in real-world applications the domain shift issue needs to be addressed. The proposed method exploits a segmentation network based on the DeepLab-v2 framework [5] that is trained using both labeled and unlabeled data in an adversarial learning framework with multiple components. The first component that controls the training is a standard cross-entropy loss exploiting ground truth annotations used to perform a supervised training on synthetic data. The second is an adversarial learning scheme inspired by previous works on semi-supervised semantic segmentation, *i.e.*, dealing with partially annotated datasets [350, 351]. We exploited a fully convolutional discriminator which produces a pixel-level confidence map distinguishing between data produced by the generator (both from real or synthetic data) and the ground truth segmentation maps. It allows to train in an adversarial setting the segmentation network using both synthetic labeled data and real-world scenes without ground truth information. Finally, the third term is based on a self-teaching loss. This key component is based on the idea introduced in [350] that the output of the discriminator can also be used as a measure of the reliability of the network estimations. This can in turn be exploited to select the reliable regions in a self-teaching framework. However, this component has been greatly improved in this work, both with respect to [350] and to [26]. First of all, the output of the discriminator has been considered as a weight to be applied to the loss function of the self-teaching component at each location, in place of the hard threshold mask used in previous work [26]. Then, a novel region growing scheme is introduced in order to extend and better represent the shape of reliable regions. This is a key difference because the previous approaches [26,350] tend to almost always discard edge regions and small objects. Finally, since the various classes have different frequencies, we also weighted the loss coming from unlabeled data in proportion to the frequency of the various classes in the synthetic dataset. This allows to obtain a better balance of the performance among the different classes. In particular, it avoids dramatic drops in performance on less common classes, as small objects and structures.

8.2.1 Preliminaries

The work of [352] has opened the way to adversarial learning approaches for the semantic segmentation task, while [258] to their application to semi-supervised learning. The approaches of [350, 351] are also based on adversarial learning but exploit a Fully Convolutional Discriminator (FCD) trying to discriminate between the predicted probability maps and the ground truth segmentation distributions at pixel-level. These works targeted a scenario where only part of the dataset is labeled but unlabeled data comes from the same dataset and shares the same domain data distribution of the labeled ones.

The work of [26] starts from [350] but instead proposes to tackle a scenario where unlabeled data refers to a different dataset with a different domain distribution, *i.e.*, it deals with the domain adaptation task. A common setting for this task is domain adaptation from synthetic data to real-world scenes. Indeed, the development of advanced computer graphics techniques enabled the collection of huge synthetic datasets for semantic segmentation. Examples of synthetic semantic segmentation datasets for the autonomous driving scenario are the GTA5 [243] and SYNTHIA [341] datasets, which have been employed in this work. However, there is a

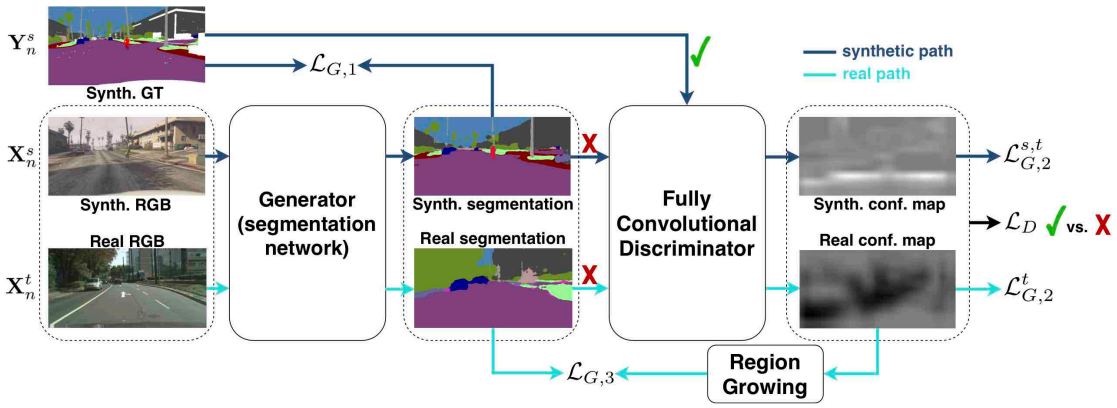


Figure 8.1: Architecture of the proposed framework. The optimization is guided by a discriminator loss and 3 losses for the generator: a standard cross-entropy loss on synthetic data ($\mathcal{L}_{G,1}$), an adversarial loss ($\mathcal{L}_{G,2}^{s,t}$) and a self-teaching loss for unlabeled real data ($\mathcal{L}_{G,3}$).

cross-domain shift that has to be addressed when a neural network trained on synthetic data processes real-world images, since in this case training and test data are not drawn i.i.d. from the same underlying distribution as usually assumed [326, 353–356].

Region growing is a long-standing problem in image segmentation methods being an unsupervised approach that examines neighboring pixels of initial seed points and determines whether the pixel neighbors should be added to the seed region depending on a region similarity criterion. Such techniques have been recently applied to domain adaptation in semantic segmentation [248, 260]. In particular in [260] a semantic segmentation network is trained to segment the discriminative regions first and to progressively increase the pixel-level supervision by seeded region growing [357]. In [248] the authors propose a saliency guided weakly supervised segmentation network which utilizes salient information as guidance to help weakly segmentation through a seeded region growing procedure. In [358] the region growing problem is represented as a Markov Decision Process.

8.2.2 Architecture of the Proposed Approach

Our target is to train a semantic segmentation network in a supervised way on synthetic data and to adapt it in a unsupervised way to real data. In this work, we name this network G, since it has the role of the *generator* in the proposed adversarial training framework. A supplementary discriminator network D is used to evaluate the reliability of G's output. This information can be employed to guide the adaptation of G to unlabeled real data. In this section, we detail the CNN architectures and the training procedure implementing the unsupervised domain adaptation. Our approach is agnostic to the architecture of G and in general any semantic segmentation network can be used. However, in our experiments G is a DeepLab-v2 network [5]. This widely used model is based on the ResNet-101 backbone whose weights were pre-trained [119] on the MSCOCO dataset [120].

Figure 8.1 shows the architecture of the proposed training framework. The optimization of the network is driven by the minimization of three loss functions. The first loss function ($\mathcal{L}_{G,1}$) is a standard multi-class cross-entropy. The segmentation network G is trained to estimate for each input pixel the probability that it belongs to a class c inside the set of possible classes $\mathcal{C}$. It is optimized only on labeled synthetic data since the ground truth is required. In the following,

$G(\mathbf{X}_n^s)$ is used to represent the output of the segmentation network on the n-th input image, $\mathbf{X}_n^s$, from the source (synthetic) domain. $\mathbf{Y}_n^s$ is used to refer to the one-hot encoded ground truth segmentation related to input $\mathbf{X}_n^s$. In this scenario, the multi-class cross-entropy loss $\mathcal{L}_{G,1}$ is formulated as:

$$\mathcal{L}_{G,1} = -\sum_{p \in \mathbf{X}_n^s} \sum_{c \in \mathcal{C}} \mathbf{Y}_n^{s\,(p)}[c] \cdot \log\left(G(\mathbf{X}_n^s)^{(p)}[c]\right) \tag{8.1}$$

where p is the index of a pixel in the considered image, c is a specific class belonging to $\mathcal{C}$ and $\mathbf{Y}_n^{s\,(p)}[c]$ and $G(\mathbf{X}_n^s)^{(p)}[c]$ are the values relative to pixel p and class c respectively in the ground truth and in the generator (G) output. As mentioned above, this loss can be computed only on the source (synthetic) domain where the semantic ground truth is available.

The second and the third loss functions, minimized during the training of G, aim at adapting the semantic segmentation CNN G to real data without using ground truth labels for real data. These loss functions are implemented by means of the discriminator network D, that is trained to distinguish segmentation maps produced by the generator from the ground truth ones. The peculiarity of this discriminator network is that it produces a per-pixel estimation, differently from traditional adversarial frameworks where the discriminator outputs a single binary value for the whole input image. The discriminator D is made of a stack of 5 convolutional layers each with 4×4 kernels with a stride of 2 and Leaky ReLU activation function. The number of filters (from the first layer to the last one) is 64, 64, 128, 128, 1 and the cascade is followed by a bilinear upsampling to match the original input image resolution. The discriminator is trained by minimizing the loss function $\mathcal{L}_D$, that is a standard cross-entropy loss, between D's output and the one-hot encoding indicating if the input is produced by G (class 0) or if it is the ground truth one-hot encoding semantic segmentation (class 1). $\mathcal{L}_D$ can be formulated as:

$$\mathcal{L}_D = -\sum_{p \in \mathbf{X}_n^{s,t}} \log(1 - D(G(\mathbf{X}_n^{s,t}))^{(p)}) + \log(D(\mathbf{Y}_n^s)^{(p)}) \tag{8.2}$$

Notice that the class 0, associated to G's output, can be produced both from an input $\mathbf{X}_n^s$ coming from the source domain and from a real-world input $\mathbf{X}_n^t$. This means that D can be trained on both synthetic and real data, trying to discriminate generated data from ground truth one. The segmented source and target datasets share a similar statistic, since low level features of the color images are processed to leave place to the class statistic: for this reason the training of D on real and synthetic data is possible. Another possible issue in the training procedure could be related to the well distinguishable Dirac distributed segmentation ground truth data. In principle, this could be easily distinguished from data produced by G. However, we have investigated this issue and in general G produces segmentation maps very close to the Dirac distribution after a few training steps. This forces D to capture also other statistical properties of the two different types of input data. Notice that this issue has been investigated also in [26, 350] with similar conclusions. The discriminator D is used to implement the second loss function for the training of G, $\mathcal{L}_{G,2}^{s,t}$. This loss function is an adversarial loss since G, the generator in the traditional adversarial training scheme, is updated in order to create an output that has to look similar to ground truth data from the D viewpoint. On a generic image $\mathbf{X}_n^{s,t}$ this loss function can be formulated as:

$$\mathcal{L}_{G,2}^{s,t} = -\sum_{p \in \mathbf{X}_n^{s,t}} \log(D(G(\mathbf{X}_n^{s,t}))^{(p)}) \tag{8.3}$$

As for the training of D (Eq. (8.2)), $\mathcal{L}_{G,2}^{s,t}$ can be optimized both on the source and on the

target data. In case the input is coming from the source dataset, we will refer to the loss function of Eq. (8.3) with $\mathcal{L}^s_{G,2}$, otherwise we will refer to it with $\mathcal{L}^t_{G,2}$ in case of target data as input. Notice that the generator is forced to adapt to the target real domain in an unsupervised way by minimizing $\mathcal{L}^t_{G,2}$. G is forced to produce data similar to what D considers ground truth also on real data. Remember that the ground truth is not used for this loss.

The third loss function is inspired from the work of Hung *et al.* [350]. The idea is to interpret the output of the discriminator D as a measure of the reliability of the output of G in case of synthetic and real data. This reliability measure is used to realize a self-training on real data. The predictions of G, assumed to be reliable by D, are converted to the one-hot encoding and are used as a self-taught ground truth to train G on unlabeled target real data. This loss can be formulated as

$$\mathcal{L}_{G,3} = -\sum_{p \in \mathbf{X}^t_n} \sum_{c \in \mathcal{C}} D_R(\mathbf{X}^t_n)^{(p)} \cdot W^s_c \cdot \hat{\mathbf{Y}}^{(p)}_n[c] \cdot \log\big(G(\mathbf{X}^t_n)^{(p)}[c]\big) \tag{8.4}$$

where $\hat{\mathbf{Y}}_n$ is the one-hot encoded ground truth derived from the per-class argmax of the generated probability map $G(\mathbf{X}_n)$. Each contribution to the loss is weighted by two terms. The first (D_R) is a weighting term dependent on the output of the discriminator refined by a region growing procedure that exploits pixel aggregation to improve the confidence estimation. The second (W^s_c) is a weighting function proportional to the class frequencies on the source domain.

More in detail, the first term finds the reliable locations in the segmented map and assigns to them a weight interpreted as a confidence measure. The module computing this weighting mask is named $D_R(\cdot)$ and it takes as input a real image $\mathbf{X}^t_n$. In the first step, a mask m_{T_u} is computed selecting confident points by applying a threshold T_u to the output of the discriminator with input $G(\mathbf{X}^t_n)$. The discriminator output is interpreted as a confidence map related to the segmentation map estimated on $\mathbf{X}^t_n$ in this phase. Formally, at each pixel location p we have:

$$m^{(p)}_{T_u} = \begin{cases} 1 & \text{if } D(G(\mathbf{X}^t_n))^{(p)} > T_u \\ 0 & \text{otherwise} \end{cases} \tag{8.5}$$

In the second step, for a generic confident pixel p in m_{T_u}, assigned by G to class c^*, the algorithm expands the confident region to a generic adjacent pixel $p' \in \mathbf{X}^t_n$ if the output of the segmentation network for the class c^* (*i.e.*, the one selected for point p) is greater than a threshold $T_{\mathcal{R}}$ at location p'. More formally, p' is added to the mask if $G(\mathbf{X}^t_n)^{(p')}[c^*] > T_{\mathcal{R}}$. We will denote with $m^R_{T_u}$ the mask obtained by applying this region growing process to the original mask m_{T_u}. Finally, for each location p^R selected by the updated mask $m^R_{T_u}$ the weight is given by the corresponding output of the discriminator $D(G(\mathbf{X}^t_n))^{(p^R)}$. Thus, the resulting weights $D_R(\mathbf{X}^t_n)$ are:

$$D_R(\mathbf{X}^t_n) = m^R_{T_u} \cdot D(G(\mathbf{X}^t_n)) \tag{8.6}$$

i.e., the weight is equal to the discriminator output for points selected by $m^R_{T_u}$ and to 0 for points not selected by the mask. Empirically we set $T_u = 0.2$ and $T_{\mathcal{R}} = 1 - 10^{-5}$ thus achieving high reliability when expanding the confidence map.

The second weighting function is related to the class frequency on the source domain (W^s_c). It is defined as:

$$W^t_c = 1 - \frac{\sum_n |p \in \mathbf{X}^s_n \wedge p \in c|}{\sum_n |p \in \mathbf{X}^s_n|}, \tag{8.7}$$

where $|\cdot|$ represents the cardinality of the considered set.

This weighting function balances the overall loss when unlabeled data of the target set are

used, avoiding that rare and tiny objects (*e.g.*, *traffic lights* or *pole*) are forgotten and replaced by more frequent and large ones (such as *road*, *building*). Notice that W_c^s is estimated on source data since the ground truth of the target data is assumed to be unknown during the training phase. Furthermore, W_c^s does not change during the training process and so it is computed only once.

Finally, the overall loss function for the training of G is a weighted average of the three losses, *i.e.*:

$$\mathcal{L}_{full} = \mathcal{L}_{G,1} + w^{s,t}\mathcal{L}_{G,2}^{s,t} + w'\mathcal{L}_{G,3} \tag{8.8}$$

We empirically set the weighting parameters as specified in Section 8.2.3. The discriminator is trained minimizing $\mathcal{L}_D$ (Eq. (8.2)) on ground truth labels and on the generator output computed on a mixed batch composed by both source and target data. During the first 5000 steps, the loss $\mathcal{L}_{G,3}$ is disabled, setting $w' = 0$, allowing the discriminator to learn how to produce higher quality confidence maps before using them. After this initial phase, all the three components of the loss are enabled and the training ends after 20000 steps.

8.2.3 Experimental Results

The target of the proposed approach is to adapt a deep network trained on synthetic data to real-world scenes. To evaluate the performance on this task we used the 4 different datasets introduced in Section 7.4. The evaluation scenario is the same of recent competing approaches as [273,326,359] in order to allow for a fair comparison. During the training stage all the images have been resized and cropped to 750×375 px for memory constraints. The testing on the real datasets, instead, has been carried out at their original resolution.

We started by evaluating the performance on the validation set of Cityscapes. In the first experiment, we trained the network using the scenes from the GTA5 dataset to compute the supervised loss $\mathcal{L}_{G,1}$ and the adversarial loss $\mathcal{L}_{G,2}^s$ while the training scenes of the Cityscapes dataset have been used for the unsupervised domain adaptation, *i.e.*, to compute the losses $\mathcal{L}_{G,2}^t$ and $\mathcal{L}_{G,3}$. Notice that no labels from the Cityscapes training set have been used. In the second experiment, we performed the same procedure but we replaced the GTA5 dataset with the SYNTHIA one.

Then, we switched to the Mapillary dataset and we repeated the two experiments using this dataset: we performed the supervised training with GTA5 or SYNTHIA and we used the training set of Mapillary, without any label, for the unsupervised domain adaptation. Similarly to the previous scenario, we evaluated the results on the validation split of Mapillary.

8.2.3.1 Training Details

The proposed deep learning scheme has been implemented using the TensorFlow framework [125]. The generator network G (we used a DeepLab-v2 model) has been trained as proposed in [5] using the Stochastic Gradient Descent (SGD) optimizer with momentum set to 0.9 and weight decay to 10^{-4}. The discriminator D has been trained using the Adam optimizer. The learning rate employed for both G and D started from 10^{-4} and was decreased up to 10^{-6} by means of a polynomial decay with power 0.9. We trained the two networks for $20,000$ iterations on a NVIDIA GTX 1080 Ti GPU. The longest training inside this work, *i.e.*, the one with all the loss components enabled, takes about 20 hours to complete. Further resources and the code of our approach are available at: `https://lttm.dei.unipd.it/paper_data/semanticDA`.

We assess the quality of our approach computing the mean Intersection over Union (mIoU), as done by all competing approaches. Moreover, we compared our results with some recent frameworks [26, 273, 326, 350].

GTA5 → Cityscapes	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mean
Supervised ($\mathcal{L}_{G,1}$ only)	45.3	20.6	50.1	9.3	12.7	19.5	4.3	0.7	81.9	21.1	63.3	52.0	1.7	77.9	26.0	39.8	0.1	4.7	0.0	27.9
Ours (full)	81.0	19.6	65.8	**20.7**	**12.9**	**20.9**	6.6	0.2	82.4	**33.0**	**68.2**	**54.9**	6.2	80.3	28.1	41.6	**2.4**	8.5	0.0	**33.3**
Hoffman *et al.* [273]	70.4	**32.4**	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
Hung *et al.* [350]	**81.7**	0.3	68.4	4.5	2.7	8.5	0.6	0.0	**82.7**	21.5	67.9	40.0	3.3	**80.7**	**34.2**	**45.9**	0.2	8.7	0.0	29.0
Zhang *et al.* [326]	74.9	22.0	**71.7**	6.0	11.9	8.4	**16.3**	**11.1**	75.7	13.3	66.5	38.0	**9.3**	55.2	18.8	18.9	0.0	**16.8**	**14.6**	28.9
Biasetton *et al.* [26]	54.9	23.8	50.9	16.2	11.2	20.0	3.2	0.0	79.7	31.6	64.9	52.5	7.9	79.5	27.2	41.8	0.5	10.7	1.3	30.4

Table 8.1: mIoU on the different classes of the Cityscapes validation set. The approaches have been trained in a supervised way on the GTA5 dataset and the unsupervised domain adaptation has been performed using the Cityscapes training set. The highest value in each column is highlighted in bold.

SYNTHIA → Cityscapes	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	sky	person	rider	car	bus	mbike	bike	mean
Supervised ($\mathcal{L}_{G,1}$ only)	10.3	20.5	35.5	1.5	0.0	**28.9**	0.0	1.2	83.1	74.8	**53.5**	7.5	65.8	18.1	**4.7**	1.0	25.4
Ours (full)	**80.7**	0.3	**75.0**	0.0	0.0	19.5	0.0	0.4	84.0	**79.4**	46.6	0.8	**80.8**	**32.8**	0.5	0.5	**31.3**
Hoffman *et al.* [273]	11.5	19.6	30.8	**4.4**	0.0	20.3	0.1	**11.7**	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6	20.1
Hung *et al.* [350]	72.5	0.0	63.8	0.0	0.0	16.3	0.0	0.5	**84.7**	76.9	45.3	1.5	77.6	31.3	0.0	0.1	29.4
Zhang *et al.* [326]	65.2	**26.1**	74.9	0.1	**0.5**	10.7	**3.7**	3.0	76.1	70.6	47.1	**8.2**	43.2	20.7	0.7	**13.1**	29.0
Biasetton *et al.* [26]	78.4	0.1	73.2	0.0	0.0	16.9	0.0	0.2	84.3	78.8	46.0	0.3	74.9	30.8	0.0	0.1	30.2

Table 8.2: mIoU on the different classes of the Cityscapes validation set. The approaches have been trained in a supervised way on the SYNTHIA dataset and the unsupervised domain adaptation has been performed using the Cityscapes training set. The highest value in each column is highlighted in bold.

8.2.3.2 Evaluation on the Cityscapes Dataset

We started the experimental evaluation from the Cityscapes dataset. Table 8.1 refers to the first experiment (*i.e.*, using GTA5 for the supervised training). It shows the accuracy obtained with standard supervised training, with the proposed approach and with some state-of-the-art approaches. By simply training the network in a supervised way on the GTA5 dataset and then performing inference on real-world data from the Cityscapes dataset a mIoU of 27.9% can be obtained. The proposed unsupervised domain adaptation strategy allows to enhance the accuracy to 33.3% with an improvement of 5.4%. By looking more in detail to the various class accuracies, it is possible to see that the accuracy has increased on almost all the classes (only on two of them the accuracy has slightly decreased). In particular, there is a large improvement on the most common classes corresponding to large structures, since the domain adaptation strategy allows to better learn their statistics in the new domain. At the same time the performance improves also on less frequent classes corresponding to small objects due to the usage of the class weights W_c^t in the self-teaching loss component.

The method of Hung *et al.* [350], based on a similar framework, achieves a lower accuracy of 29% mostly because it struggles with small structures and uncommon classes. The methods in [273, 326] also have lower performance; however, they are also based on a different generator network. The older version of our method, introduced in [26], achieves an accuracy of 30.4%, with a gap of almost 3% with respect to the proposed approach, proving that the newly introduced elements (*i.e.*, the weighting in the self-teaching and the region growing strategy) have a relevant impact on the performance.

Figure 8.2a shows the output of the supervised training, of the methods of [350] and [26] and of our approach on some sample scenes, using the GTA5 dataset as source dataset and the Cityscapes as target one. The supervised training leads to reasonable results, but some small objects get lost or the object contours are badly captured (*e.g.*, the rider in row 1 or the poles in row 3). Furthermore, some regions of the street are corrupted by noise (*e.g.*, see rows 1 and

2). The approach of [350] seems to lose some structures (*e.g.*, the terrain in the third row) and presents issues with small objects (the poles in row 3 get completely lost) as pointed out before. The old version of the approach [26] has better performance, for example the people are better preserved and the structures have better defined edges but there are still artifacts, *e.g.*, the road surface in row 2 and 3. Finally, the proposed method has the best performance showing a good capability of detecting small objects and structures and at the same time a reliable recognition of the road and of the larger elements in the scene: in all the selected images it obtains a cleaner representation of the road removing the *sidewalk* class where is not present but at the same time correctly localizes it in the second row differently from the other methods. Similar discussion holds for the *terrain* class in row 3 and for the *pole* class whose detection has been highly improved with respect to [350].

Adapting from SYNTHIA, the task is even more challenging with respect to the GTA5 case since the computer generated graphics are less realistic. By training the network G in a supervised way on SYNTHIA and performing inference on the real-world Cityscapes dataset, a mIoU of 25.4% can be obtained (see Table 8.2). This value is smaller than the mIoU of 27.9% obtained by training G on the GTA5 dataset. The performance gap confirms that the GTA5 dataset has a smaller domain shift with respect to real-world data, when compared with the SYNTHIA dataset. By exploiting the proposed approach an accuracy of 31.3% can be obtained. The improvement is very similar to the one obtained using GTA5 as source dataset, proving that the approach is able to generalize to different datasets. In this case, there is a larger variability among different classes, however notice the very large improvement on *road* and *building* classes. The previous version of the method [26] has an accuracy of 30.2%

Furthermore, our framework outperforms the compared state-of-the-art approaches. The method of Hung *et al.* [350], that exploits the same generator architecture of our approach, obtains a mIoU equal to 29.4%. The approach of [326] has an even lower mIoU of 29.0%. The method of [273] is the less performing approach and in this comparison it is even less accurate than our synthetic supervised trained network, however it employs a different segmentation network.

The fourth, fifth and sixth row of Figure 8.2a shows the output on the same sample scenes discussed above when the SYNTHIA dataset is used as source instead of GTA5. The first thing that stands out is that by training on the SYNTHIA dataset some very common classes as *sidewalk* and *road* are highly corrupted. This is caused by the not very realistic textures used for such classes in the SYNTHIA dataset. Furthermore, while the positioning of the camera in the Cityscapes dataset is always fixed and mounted on-board inside the car, in SYNTHIA the camera can be placed in different positions. For example, the pictures can be captured from inside the car, from the top or from the side of the road.

The approach of Hung *et al.* [350] is able to correctly recognize the class *road*, correcting the noise present in the synthetic supervised training. However, as mentioned earlier, it suffers on small classes where it tends to lose small objects and to produce imprecise shapes. The method of [26] and the proposed one have slightly better performance: the last two columns of Figure 8.2a show how the unsupervised adaptation and the self-teaching component allow to avoid all the artifacts on the *road* surface. The segmentation network now captures the real nature of this class in the Cityscapes dataset. At the same time, our method is able to locate a bit more precisely small classes as *person* and *vegetation*. However, in this setting the difference between the old and new version of the proposed method is limited.

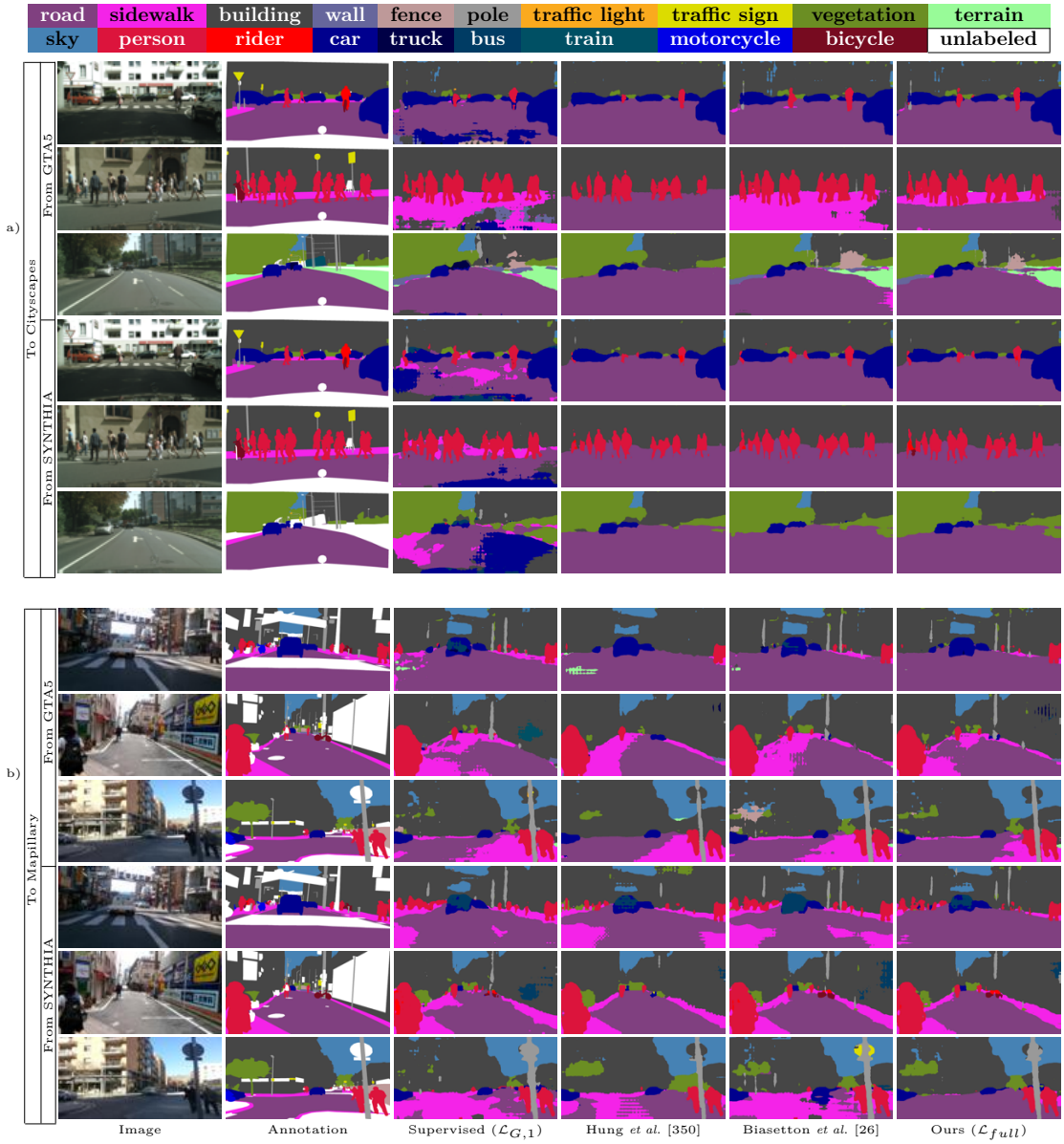


Figure 8.2: Semantic segmentation of some sample scenes extracted from the Cityscapes (a) and Mapillary (b) validation sets. The first group of six rows is related to the Cityscapes dataset, the last six to the Mapillary dataset. For each group, the first three rows are related to the experiments in which the GTA5 dataset is used as source. The last three rows are related to the case in which the SYNTHIA dataset is used as source.

GTA5 → Mapillary	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mean
Supervised ($\mathcal{L}_{G,1}$ only)	66.5	24.4	46.1	17.9	21.6	24.8	**11.8**	**5.9**	70.7	25.6	66.1	57.3	10.2	79.7	37.3	39.8	4.6	10.1	**1.7**	32.7
Ours (full)	**79.9**	28.0	**73.4**	**23.0**	**29.5**	20.9	1.1	0.0	**79.5**	**39.6**	95.0	57.6	9.0	**80.6**	**41.5**	40.1	**7.4**	**24.8**	0.1	**38.5**
Hung *et al.* [350]	78.2	**29.7**	68.7	10.0	6.7	17.5	0.0	0.0	76.4	35.2	**95.6**	53.8	13.8	77.5	34.3	30.2	5.0	21.8	0.0	34.4
Biasetton *et al.* [26]	71.4	25.0	62.0	20.4	17.6	**26.8**	5.9	0.8	64.6	24.6	86.5	**58.3**	**14.7**	80.0	39.3	**42.2**	5.5	22.3	0.1	35.2

Table 8.3: mIoU on the different classes of the Mapillary validation set. The approaches have been trained in a supervised way on the GTA5 dataset and the unsupervised domain adaptation has been performed using the Mapillary training set. The highest value in each column is highlighted in bold.

SYNTHIA → Mapillary	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	sky	person	rider	car	bus	mbike	bike	mean
Supervised ($\mathcal{L}_{G,1}$ only)	14.7	18.6	34.6	**5.4**	**0.1**	28.5	**0.0**	0.4	73.8	62.9	50.0	**11.4**	**74.3**	**28.7**	**14.0**	8.1	26.6
Ours (full)	**57.6**	18.3	**62.1**	0.4	0.0	23.7	**0.0**	0.0	**79.4**	94.8	**52.4**	9.2	74.2	28.3	4.0	6.9	**32.0**
Hung *et al.* [350]	36.8	**20.1**	53.9	0.0	0.0	23.7	**0.0**	0.0	73.9	**95.6**	43.4	0.1	64.6	19.0	0.4	0.5	27.0
Biasetton *et al.* [26]	16.4	19.1	42.2	2.7	0.0	**33.1**	**0.0**	**1.3**	76.5	88.0	50.4	10.9	69.9	25.5	6.1	**9.2**	28.2

Table 8.4: mIoU on the different classes of the Mapillary validation set. The approaches have been trained in a supervised way on the SYNTHIA dataset and the unsupervised domain adaptation has been performed using the Mapillary training set. The highest value in each column is highlighted in bold.

8.2.3.3 Evaluation on the Mapillary dataset

To ensure that our approach can generalize to other real datasets, we performed the same experimental evaluation procedure also on the Mapillary dataset (reported in Tables 8.3 and 8.4). We started by using the GTA5 dataset for the supervised training as before. By simply performing a supervised training on GTA5 and then testing on the Mapillary dataset a mIoU of 32.7% can be obtained. The proposed approach allows to obtain a much more accurate classification with a mIoU of 38.5%. Notice that the gain of almost 6% is consistent with the results obtained on the Cityscapes dataset, proving that the performance of the approach is stable across different datasets. The improvement can also be appreciated on both small and large classes, the mIoU values of 14 out of 19 classes show a clear gain. This is also visible in the qualitative results depicted in Figure 8.2b, where most of the artifacts on the road surface present in the synthetic trained network disappear and the shape of the small objects is more accurate. The results of [273] and [326] are not available for this dataset, however notice how the approach outperforms by a large margin both [350] and the old version of the approach [26] that are able to reduce only partially the artifacts on the road surface (visible in all the images), on the cars (row 1) and on the buildings (row 3).

Furthermore, we can appreciate that also on Mapillary the accuracy is lower when adapting from SYNTHIA leading to a mIoU of 26.6% only. As for Cityscapes, *road* and *sidewalk* classes have an extremely low accuracy due to the poor texture representation (the visual results are reported in the last 3 rows of Figure 8.2b). By exploiting the proposed unsupervised domain adaptation strategy the mIoU increases to 32.0% with an improvement of 5.4%, again consistent with the other experiments. In this case, the performance is more unstable across the various classes but it is noticeable the large gains on *road* and *building* classes. This is also confirmed by the qualitative results, for example we can appreciate that the proposed approach is the only one able to achieve an accurate and reliable recognition of the road. The method of Hung *et al.* [350] achieves a mIoU of 27% with a very limited improvement with respect to the synthetic supervised training. It is strongly penalized by the poor performance on the small and uncommon classes. The approach of [26] has slightly better performance (28.4%), but it has a quite large

$\mathcal{L}_{G,1}$	$\mathcal{L}_{G,2}$	$\mathcal{L}_{G,3}$	Region Growing	Disc. Weight.	Class Weight. (W_c^t)	mIoU GTA	mIoU SYNTHIA
✓						27.9	25.4
✓	✓					29.4	27.4
✓	✓	✓				30.4	30.2
✓	✓	✓	✓		✓	32.6	31.0
✓	✓	✓		✓	✓	32.8	30.2
✓	✓	✓	✓	✓		33.1	30.2
✓	✓	✓	✓	✓	✓	**33.3**	**31.3**

Table 8.5: Ablation study on Cityscapes validation set. We analyze the influence of the losses $\mathcal{L}_{G,1}$, $\mathcal{L}_{G,2}$, $\mathcal{L}_{G,3}$, and of the strategies of region growing, discriminator weighting and class weighting W_c^t.

gap with respect to the proposed method. The weighting scheme and the region growing strategy introduced in this work allowed to obtain a large improvement in this setting.

8.2.3.4 Ablation Study

In this section, we are going to analyze the contributions of the various components of the proposed approach. We focus on Cityscapes as target dataset for this study. The results of this analysis are shown in Table 8.5. By training the generator with a synthetic supervised approach, *i.e.*, using only $\mathcal{L}_{G,1}$, it is possible to obtain a mIoU of 27.9% when GTA5 is the source dataset and 25.4% when adapting from SYNTHIA. As mentioned in the previous sections, this is the less performing approach. A slight improvement can be obtained by adding the adversarial term $\mathcal{L}_{G,2}$ in the loss function. In this case, the mIoU increases of 1.5% and 2% when the source datasets are GTA5 and SYNTHIA respectively. The use of the self-teaching loss $\mathcal{L}_{G,3}$ is particularly useful when exploiting the SYNTHIA dataset, obtaining an improvement of almost 3%, probably because the domain shift from this dataset is larger. In the case of GTA5, the improvement is smaller but still significant. Moving to the new elements introduced in this work, the region growing strategy (*i.e.*, masking with the extended mask $m_{T_u}^R$ instead of using m_{T_u}) allows to a further performance enhancement, especially when using GTA5 with a 2.2% increase, mostly due to the improved handling of medium and large size objects. When starting from SYNTHIA the gain is more limited but still noticeable (almost 1%). The usage of the discriminator output as a weighting factor for the self-teaching loss, without masking with $m_{T_u}^R$, has a more unstable behavior. This leads to a very good improvement of 2.4% when starting from GTA5 but having almost no impact when employed alone in the SYNTHIA case. Moreover, we can observe that removing the class weighting term W_t^c in Eq. (8.4), *i.e.*, assigning the same weight to all the classes, we obtain a mIoU of 33.1% and 30.2% when adapting from GTA5 and SYNTHIA, respectively. The values are not too far from the mIoU of the complete version of the approach (33.3% and 31.3%, respectively). This proves that the performance are quite stable with respect to the setting of the weights for the various classes: the approach can work well even if the class frequencies on real data do not accurately match the statistics of synthetic data. Notice how the complete version of our approach has the best performance. In particular the discriminator-based weighting on the SYNTHIA dataset, that alone had a limited impact, is useful when combined with the region growing scheme.

Finally, Table 8.6 analyzes the impact of the weights that control the relative relevance of the 3 losses. It is possible to notice that the most critical parameter is the weight of the adversarial loss on the source domain w^s, that has the largest impact on the final accuracy while the performance

w^s	w^t	w'	mIoU from GTA5	mIoU from SYNTHIA
$\times\mathbf{1}$	$\times\mathbf{1}$	$\times\mathbf{1}$	**33.3**	**31.3**
$\times 1$	$\times 1$	$\times 10$	32.5	29.8
$\times 1$	$\times 1$	$\times 0.1$	32.4	29.2
$\times 1$	$\times 10$	$\times 1$	31.0	28.8
$\times 1$	$\times 0.1$	$\times 1$	30.8	24.4
$\times 10$	$\times 1$	$\times 1$	27.2	23.3
$\times 0.1$	$\times 1$	$\times 1$	30.8	23.4
$\times 4$	$\times 1$	$\times 1$	29.1	29.0
$\times 0.25$	$\times 1$	$\times 1$	32.5	25.2
$\times 2$	$\times 1$	$\times 1$	30.1	30.5
$\times 0.5$	$\times 1$	$\times 1$	32.3	28.1

Table 8.6: Ablation study on the Cityscapes validation set when adapting from GTA5. Different values of the balancing hyper-parameters of Eq. (8.8) are reported applying various scaling factors to each parameter. The default parameters are $w^s = 10^{-2}, w^t = 10^{-4}, w' = 10^{-3}$ when adapting from GTA5 and $w^s = 10^{-2}, w^t = 10^{-3}, w' = 10^{-1}$ when adapting from SYNTHIA.

are more stable with respect to the other two parameters.

8.3 UDA with Adversarial Learning with Multiple Discriminators

In the second contribution [26,27], we move from the structure proposed in the previous section. However, in this work [28] we use two discriminative networks, one to differentiate between ground truth and predicted segmentations and the other to discriminate between segmentation maps coming from synthetic and real-world data. Finally, a self-training component is introduced based on the idea that the output of the discriminator provides a measure of the reliability of the network estimations to be exploited in a self-training framework [26,350]. In previous works, this module lacked two fundamental aspects: it was not class-wise adaptive and was not able to update the thresholding parameters during training. In other words, different classes shared the same confidence threshold to select regions for self-training, and its value was kept fixed during all the learning process. In this work, instead, we introduce an adaptive thresholding scheme for the selection of the regions used for self-training that enforces a balanced selection for all classes and allows for variability over different training steps. Hence, the framework can accomplish both inter-class confidence flexibility and time adaptability throughout the optimization phase.

In summary, the main contributions of this work are: (1) we introduce a novel adversarial scheme exploiting multiple domain discriminators to align the source and target domains, (2) we design a self-training module with adaptive confidence both over different classes and over different training steps, (3) we prove the effectiveness of our framework on 4 different experimental scenarios outperforming competing approaches.

8.3.1 Proposed Domain Adaptation Strategy

The complete architecture of the proposed approach is now discussed. An overview of the method is given in Figure 8.3. We denote with G the semantic segmentation network that we want to adapt from supervised synthetic data to unlabeled real data. G takes the role of the generator in our adversarial setup. The optimization of the segmentation network is driven

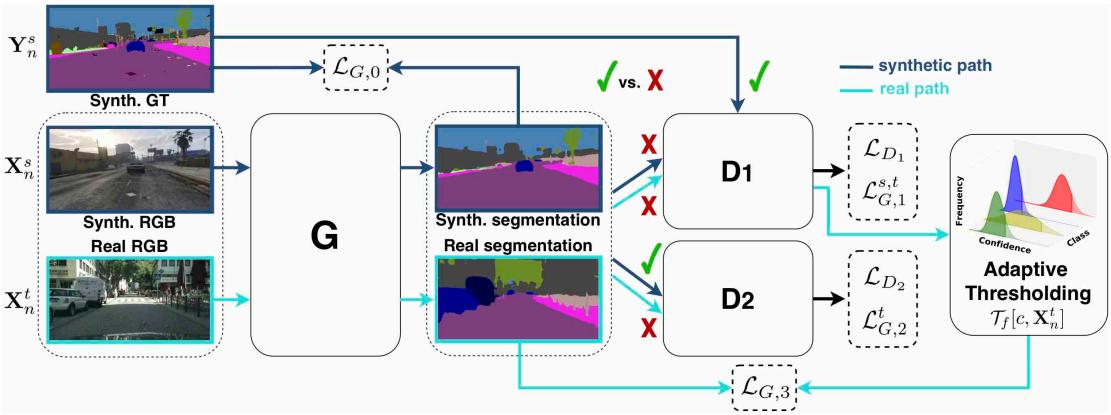


Figure 8.3: Architecture of the proposed approach. The semantic segmentation network G is trained with the combination of 4 losses: a supervised cross entropy on source data $\mathcal{L}_{G,0}$, a double adversarial framework $\mathcal{L}_{G,1}^{s,t}$ and $\mathcal{L}_{G,2}^{t}$, and a self-training module $\mathcal{L}_{G,3}$ with class-wise and time-varying adaptive thresholding mask $\mathcal{T}_f$.

by the minimization of a multi-target objective involving four loss functions. In particular, to guide the adaptation of G to unlabeled real data we employ two discriminative networks and a self-training module.

Let us denote with $\mathbf{X}_n^s$ the generic n-th image in the source (synthetic) domain and with $\mathbf{Y}_n^s$ the corresponding ground truth segmentation, while $\mathbf{X}_n^t$ is the n-th real-world sample (for which ground truth is not available during training). The first component of our approach, $\mathcal{L}_{G,0}$, is a standard cross-entropy loss working on labeled synthetic data.

The second and the third loss functions, minimized during the training of G, are relative to the adversarial training with the couple of discriminator networks. The first discriminator module D_1 is trained to distinguish between ground truth and generated maps (the latter either coming from synthetic or real images). The peculiarity of this network is that it is a fully convolutional model and it produces a per-pixel confidence estimation, differently from traditional adversarial frameworks where the discriminator outputs a single binary value for the whole input image. The key idea is that the discriminative action provides a measure of discrepancy between the statistic of source annotations and both source and target prediction maps, and its optimization leads to an indirect yet effective cross-domain distribution alignment [26, 27, 350].

The discriminator network D_1 is made of a stack of 5 convolutional layers each using 4×4 kernels with a stride of 2 and Leaky ReLU activation function. The number of filters (from first to last layer) is 64, 64, 128, 128, 1 and the cascade is followed by a bilinear upsampling to match the original input image resolution. The network D_1 is trained to minimize the loss $\mathcal{L}_{D_1}$ that is a standard cross-entropy loss between D_1's output and the one-hot encoding indicating whether the input segmentation map is the ground truth (class 1) or it is produced by G (class 0), *i.e.*:

$$\mathcal{L}_{D_1} = -\sum_{p \in \mathbf{X}_n^{s,t}} \log(1 - D_1(G(\mathbf{X}_n^{s,t}))^{(p)}) + \log(D_1(\mathbf{Y}_n^s)^{(p)}) \tag{8.9}$$

where p is a generic pixel in the image. Meanwhile, G is forced to produce segmentation maps that resemble ground-truth annotations when inspected by D_1:

$$\mathcal{L}_{G,1}^{s,t} = -\sum_{p \in \mathbf{X}_n^{s,t}} \log(D_1(G(\mathbf{X}_n^{s,t}))^{(p)}) \tag{8.10}$$

The second discriminator, D_2, is trained to differentiate between segmentation maps coming from synthetic or real-world data. Differently from D_1, D_2 is always fed with the generator output, *i.e.* with source or target images segmented by G, whereas no ground-truth information is employed. D_2 has the same structure as D_1 except that its number of channels has been reduced to 48, 48, 96, 96, 1, as its adaptation objective is complementary to the one of D_1 and requires less computational complexity to be accomplished. The adversarial loss for D_2 can be expressed as:

$$\mathcal{L}_{D_2} = -\sum_{p \in \mathbf{X}_n^{s,t}} \log(1 - D_2(G(\mathbf{X}_n^t))^{(p)}) + \log(D_2(G(\mathbf{X}_n^s))^{(p)}) \tag{8.11}$$

which is a standard cross-entropy loss similar to $\mathcal{L}_{D_1}$. Here the objective is formulated between D_2's output and the one-hot encoding marking with 0 and 1 the segmentation maps produced by G from respectively target and source images. The role of G in the second adversarial competition is to fool D_2, by computing sufficiently realistic target prediction maps resembling the source domain ones. Hence, similarly to Eq. (8.10), the adversarial loss for G related to D_2 is of the form:

$$\mathcal{L}_{G,2}^t = -\sum_{p \in \mathbf{X}_n^t} \log(D_2(G(\mathbf{X}_n^t))^{(p)}) \tag{8.12}$$

The second adversarial framework is specifically focused on the adaptation of target network representations, in that G is trained to produce target segmentation maps which are close to source ones from a statistical point of view. Thus it directly tackles the domain gap causing the performance drop on the target set. Source-target alignment also happens within the first adversarial scenario as a side effect. Indeed, G's output is forced to be distributed as ground-truth labels for both source and target inputs, but no actual cross-domain adaptation of network embeddings is directly performed.

The last module in our adaptation framework implements a self-training strategy. As shown by recent works [26,27,350], the output of the discriminator D_1 can be interpreted as a measure of confidence for predicted target pixels, since D_1 should identify target predictions that deviate from source ground-truth statistic. Hence, the output of D_1 can be exploited to select the reliable predictions in the target segmentation maps. The selected output samples are then converted into one-hot encoded data, which can be used as a self-taught ground-truth to train G directly on unlabeled target samples. The self-training objective is expressed by the following loss:

$$\mathcal{L}_{G,3} = -\sum_{p \in \mathbf{X}_n^t} \sum_{c \in \mathcal{C}} \mathcal{M}_f^{(p)} \cdot W_c^s \cdot \hat{\mathbf{Y}}_n^{(p)}[c] \cdot \log\big(G(\mathbf{X}_n^t)^{(p)}[c]\big) \tag{8.13}$$

where $\hat{\mathbf{Y}}_n$ is the one-hot encoded ground truth derived from the per-class argmax of the generated probability map $G(\mathbf{X}_n^t)$, c is a specific class belonging to the set of possible classes $\mathcal{C}$ and W_c^s is a weighting function proportional to the class frequency on the source domain as introduced in [26, 27]. $\mathcal{M}_f^{(p)}[c, \mathbf{X}_n^t]$ (for ease of notation we drop its dependencies on class and training sample in all the equations) is the adaptive thresholding mask for the selection of the regions used for self-training, defined as:

$$\mathcal{M}_f^{(p)} = \begin{cases} 1 & \text{if } \big(D_1(G(\mathbf{X}_n^t))^{(p)} > \mathcal{T}_f[c, \mathbf{X}_n^t]\big) \wedge \Big(\hat{\mathbf{Y}}_n^{(p)}[c] = 1\Big) \\ 0 & \text{otherwise} \end{cases} \tag{8.14}$$

Differently from previous works [26,27,324,331], in which the confidence threshold was computed before-hand and kept fixed throughout the training, here we propose to have both a class-level

and a training-stage adaptation of the threshold. Indeed, different classes typically have different confidence values which may also vary during training. The class-wise confidence threshold selection for a generic training step and a generic class c is formulated as:

$$\mathcal{T}_f[c, \mathbf{X}_n^t] = \mathcal{Q}_f(D_1(G(\mathbf{X}_n^t)[c])) \tag{8.15}$$

where $\mathcal{Q}_f$ represents the f-th percentile. The best results are obtained with f in the range of $75 - 80\%$ to enable self-training only in regions with high confidence on target data (as will be shown in Figure 8.5). We compute $\mathcal{T}_f$ for every class at each training step by looking at network predictions on target data in the current batch. This makes the model adaptive both to the statistic of the various classes and to the different training phases. Finally, we can compute the overall loss function for G with a weighted average of the four individual losses, *i.e.*:

$$\mathcal{L}_{full} = \mathcal{L}_{G,0} + w_1^{s,t}\mathcal{L}_{G,1}^{s,t} + w_2^t\mathcal{L}_{G,2}^t + w_3\mathcal{L}_{G,3} \tag{8.16}$$

with weighting parameters $w_1^{s,t}$, w_2^t and w_3 empirically set.

8.3.2 Experimental Results

8.3.2.1 Training Details

The approach is agnostic to the deep learning architecture used for G: in principle any semantic segmentation network can be used, in our case we employ the well known DeepLab-v2 model [5] with the ResNet-101 backbone. The weights are pre-trained [119] on the MSCOCO dataset [120].

The proposed deep learning scheme is implemented in TensorFlow [125] and the code is available at `https://lttm.dei.unipd.it/paper_data/semanticDA`. The generator network G is trained as suggested in [5] using the Stochastic Gradient Descent (SGD) optimizer with momentum set to 0.9 and weight decay to 10^{-4}. The discriminators D_1 and D_2 are trained using the Adam optimizer. Following [26], the learning rate employed for both G and D_1 starts from 10^{-4} and is decreased up to 10^{-6} by means of a polynomial decay with power 0.9. As for D_2, the base learning rate is set to 10^{-4} and 5×10^{-4} for respectively the SYNTHIA and GTA5 adaptation scenarios. Following the empirical validation and previous works [26, 27, 350], the weighting parameters are set as $w_1^s = 10^{-2}$, $w_1^t = 10^{-3}$ and $w_2^t = 10^{-2}$, independently of the source and target datasets. The self-training parameter w_3 is fixed to 5×10^{-2} and 10^{-1} for respectively SYNTHIA and GTA5 cases. This is indeed the most delicate parameter to tune, as the inferior realism of the SYNTHIA dataset suggests a more cautious usage of the self-training module to avoid flawed supervision that noisy target pseudo-labels may provide. The approach is instead more stable with respect to the other weighting parameters. We train the model for $20K$ iterations on a NVIDIA RTX 2080 Ti GPU. The longest training inside this work, *i.e.*, the one with all the loss components enabled, takes about 6 hours to complete.

8.3.2.2 Evaluation on the Cityscapes Dataset

The first scenario we consider for evaluation purposes comprises the adaptation to the Cityscapes dataset from both SYNTHIA and GTA5. As done by competing approaches [273, 282, 287], the numerical performance is expressed in terms of mean Intersection over Union (mIoU) between predicted maps and ground-truth labels over the Cityscapes validation set. The per-class mIoU results of the evaluations are displayed in Table 8.7a) and 8.7b). We denote as *supervised* the naïve approach relying only on source supervision and no form of adaptation, while the numerical performance of our method as a whole is reported in the last row of each section.

			Method	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mean
a)	To Cityscapes	From GTA5	Supervised ($\mathcal{L}_{G,0}$ only)	49.3	24.4	56.4	6.5	**19.6**	25.6	**23.6**	10.1	82.7	28.5	69.9	**55.5**	4.9	80.9	18.0	33.0	1.2	15.1	0.1	31.9
			Hoffman *et al.* [273]	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
			Hung *et al.* [350]	**81.7**	0.3	68.4	4.5	2.7	8.5	0.6	0.0	82.7	21.5	67.9	40.0	3.3	80.7	**34.2**	**45.9**	0.2	8.7	0.0	29.0
			Zhang *et al.* [326]	74.9	22.0	**71.7**	6.0	11.9	8.4	16.3	**11.1**	75.7	13.3	66.5	38.0	**9.3**	55.2	18.8	18.9	0.0	**16.8**	**14.6**	28.9
			Biasetton *et al.* [26]	54.9	23.8	50.9	16.2	11.2	20.0	3.2	0.0	79.7	31.6	64.9	52.5	7.9	79.5	27.2	41.8	0.5	10.7	1.3	30.4
			Michieli *et al.* [27]	81.0	19.6	65.8	**20.7**	2.9	20.9	6.6	0.2	82.4	33.0	68.2	54.9	6.2	80.3	28.1	41.6	**2.4**	8.5	0.0	33.3
			Ours	77.7	**35.9**	67.2	18.9	12.1	**26.2**	15.9	5.9	**83.7**	**33.3**	**72.7**	53.9	4.2	**82.6**	21.5	41.1	0.1	13.9	0.0	**35.1**
b)		From SYNTHIA	Supervised ($\mathcal{L}_{G,0}$ only)	17.9	24.2	38.6	**5.0**	0.0	28.7	0.0	4.5	79.3	-	80.8	**54.0**	**8.9**	75.7	-	35.4	-	4.2	3.9	28.8
			Hoffman *et al.* [273]	11.5	19.6	30.8	4.4	0.0	20.3	0.1	**11.7**	42.3	-	68.7	51.2	3.8	54.0	-	3.2	-	0.2	0.6	20.1
			Hung *et al.* [350]	72.5	0.0	63.8	0.0	0.0	16.3	0.0	0.5	**84.7**	-	76.9	45.3	1.5	77.6	-	31.3	-	0.0	0.1	29.4
			Zhang *et al.* [326]	65.2	26.1	74.9	0.1	**0.5**	10.7	**3.7**	3.0	76.1	-	70.6	47.1	8.2	43.2	-	20.7	-	0.7	**13.1**	29.0
			Biasetton *et al.* [26]	78.4	0.1	73.2	0.0	0.0	16.9	0.0	0.2	84.3	-	78.8	46.0	0.3	74.9	-	30.8	-	0.0	0.1	30.2
			Michieli *et al.* [27]	**80.7**	0.3	**75.0**	0.0	0.0	19.5	0.0	0.4	84.0	-	79.4	46.6	0.8	80.8	-	32.8	-	0.5	0.5	31.3
			Ours	72.0	**26.6**	66.1	1.8	0.0	**30.2**	0.0	4.3	81.4	-	**82.2**	51.7	4.0	**82.2**	-	**37.9**	-	**7.7**	5.9	**34.6**
c)	To Mapillary	From GTA5	Supervised ($\mathcal{L}_{G,0}$ only)	69.8	31.8	58.8	14.6	22.3	28.3	**31.8**	**28.8**	70.0	24.4	72.4	**60.4**	16.8	80.6	36.6	34.3	**10.2**	**26.2**	**0.2**	37.8
			Hung *et al.* [350]	78.2	29.7	68.7	10.0	6.7	17.5	0.0	0.0	76.4	35.2	**95.6**	53.8	13.8	77.5	34.3	30.2	5.0	21.8	0.0	34.4
			Biasetton *et al.* [26]	71.4	25.0	62.0	20.4	17.6	26.8	5.9	0.8	64.6	24.6	86.5	58.3	14.7	80.0	39.3	42.2	5.5	22.3	0.1	35.2
			Michieli *et al.* [27]	79.9	28.0	73.4	**23.0**	29.5	20.9	1.1	0.0	**79.5**	**39.6**	95.0	57.6	9.0	80.6	41.5	40.1	7.4	24.8	0.1	38.5
			Ours	**80.0**	**43.3**	**75.4**	19.4	**29.7**	**29.6**	23.3	16.2	78.5	33.5	93.7	59.0	**20.3**	**82.2**	**44.5**	**43.4**	2.5	22.1	0.0	**41.9**
d)		From SYNTHIA	Supervised ($\mathcal{L}_{G,0}$ only)	25.4	22.0	56.4	**6.9**	**0.1**	29.4	0.0	**2.8**	72.8	-	92.1	53.7	**16.1**	75.1	-	**30.8**	-	8.6	5.8	31.1
			Hung *et al.* [350]	36.8	20.1	53.9	0.0	0.0	23.7	0.0	0.0	73.9	-	**95.6**	43.4	0.1	64.6	-	19.0	-	0.4	0.5	27.0
			Biasetton *et al.* [26]	16.4	19.1	42.2	2.7	0.0	**33.1**	0.0	1.3	76.5	-	88.0	50.4	10.9	69.9	-	25.5	-	6.1	9.2	28.2
			Michieli *et al.* [27]	57.6	18.3	62.1	0.4	0.0	23.7	0.0	0.0	**79.4**	-	94.8	52.4	9.2	74.2	-	28.3	-	4.0	6.9	32.0
			Ours	**59.0**	**28.4**	**68.6**	0.7	0.0	29.8	0.0	1.8	77.5	-	94.9	**54.6**	12.6	**76.8**	-	28.0	-	**11.2**	**14.7**	**34.9**

Table 8.7: Per-class and mean IoU on the four considered UDA scenarios. The approaches have been trained in a supervised way on the synthetic dataset and the unsupervised domain adaptation has been performed using the respective real-world training set. The results are reported on the real-world validation sets.

Using GTA5 as source domain, the simple supervised approach achieves a final mIoU of 31.8% on the target dataset. The introduction of the multi-domain adversarial learning scheme and adaptive self-training module leads to a performance increment of 3.3%, boosting the mIoU up to 35.1%. Moreover, the improvement is shared by the majority of the semantic classes. Some categories characterized by semantic similarity between them (such as *road*, *sidewalk* and *terrain*) or with appearance discrepancy across source and target domains (such as *car*, *truck* and *bus*) seem to highly benefit from the adaptation, proving that our method succeeded in bridging the domain gap. In Table 8.7 we also report results achieved by some competing approaches. It can be noticed that our strategy outperforms all of them, including those relying on simpler self-training and adversarial learning schemes (*i.e.*, [26, 27, 350]), demonstrating the efficacy of the multi-domain discrimination and of the adaptive thresholding techniques we introduced. In Figure 8.4a) we show some sample segmented images from the Cityscapes validation set. We can appreciate that our approach leads to a more precise detection of several semantic entities found in the input image. For example, the *road* and *sidewalk* classes are subject to a more accurate recognition, which supports the numerical results.

Section (b) of Table 8.7 reports the results of the adaptation from the SYNTHIA dataset. As for the GTA5 case, our adaptation strategy represents a considerable improvement over the supervised training on the source dataset. The mIoU achieved without adaptation (28.8%) is pushed up to 34.6% by our framework. The increment of almost 6% is even higher than the one achieved with the more realistic GTA5 dataset as source domain, and this proves the effectiveness of the adaptation modules we developed also in a challenging environment with a larger statistical gap across domains. For example, while texture discrepancy between Cityscapes and SYNTHIA causes the road on real-world scenes to be hardly recognized by the segmentation network in lack of adaptation, our strategy successfully provides the predictor with road detection capabilities. Figure 8.4a) includes some qualitative results. Again, we can observe the improved semantic

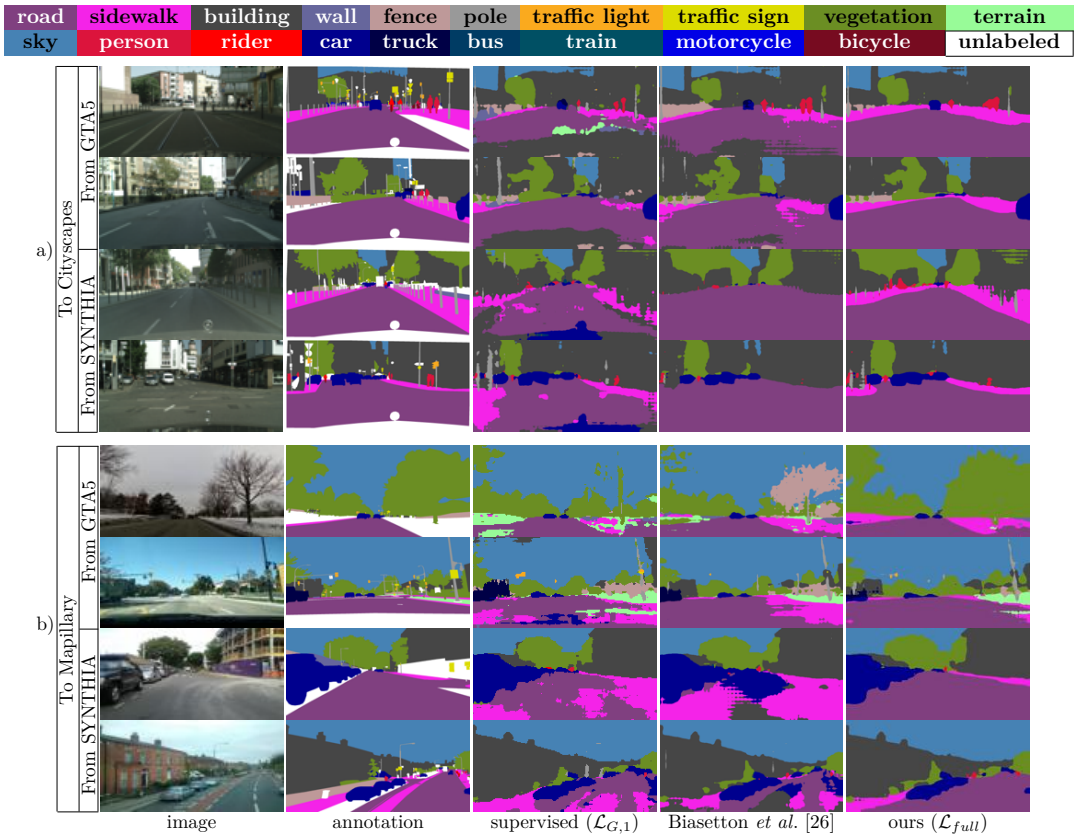


Figure 8.4: Semantic maps of sample scenes extracted from Cityscapes (first four rows, a)) and Mapillary (last four rows b)) validation sets. For each group, the first two rows are related to the experiments in which the GTA5 dataset is used as source. The last two rows are related to the case in which the SYNTHIA dataset is used as source.

understanding and detection accuracy on classes such as *road* and *sidewalk* with respect to the baseline, as well as with respect to a competing approach based on adversarial and self-training techniques [26].

8.3.2.3 Evaluation on the Mapillary dataset

We evaluate our approach also on the Mapillary dataset. We start by using the GTA5 dataset for the supervised training as before: the results are shown in Table 8.7c). With no adaptation to the Mapillary dataset the network achieves a mIoU of 37.8%. Thanks to the multiple domain discriminators and to the adaptive self-training techniques our framework is able to reach a mIoU of 41.9%, significantly outperforming all the compared methodologies. We can notice that the improvement with respect to the baseline approach is consistently distributed among the semantic classes and it is particularly evident on the *road* or *building* ones. Qualitative results are shown in the first two rows of Figure 8.4b), where we can verify that most of the noise present in the supervised training and in [26] is filtered out by the proposed framework. In particular, the *vegetation* and *sidewalk* categories highly benefit from the domain adaptation with class-wise and time-variable confidence threshold selection.

Furthermore, we can appreciate that, also on the Mapillary dataset, the accuracy is lower when the SYNTHIA dataset is used for supervised training, leading to a mIoU of 31.3% only. As already noticed from the evaluations on Cityscapes, some classes (*i.e.*, *road*, *sidewalk* and *building*) have a low accuracy due to the poor texture representation and vastly profit by the adaptation to the target domain. The complete framework increases the final mIoU to 34.9% with an improvement of 3.8%, consistent with the previous experiments made across different datasets. Remarkable are the percentage gains in the aforementioned classes: for example, the *road* class more than doubles its accuracy and *sidewalk*'s IoU grows by 12.2%. The visual results are reported in the last two rows of Figure 8.4b). Here, for example, we can appreciate that the proposed approach is the only one to achieve an accurate and reliable recognition of *road* and *cars* classes on the shown images, which confirms our previous analysis.

8.3.2.4 Ablation Study

In this section we present an accurate investigation of the effectiveness of the various modules of the proposed framework. For this study we consider the performance on the Mapillary dataset when adapting from GTA5. We start by evaluating the individual impact of each module: the performance analysis is shown in Table 8.8. Let us recall that the baseline architecture, *i.e.*, the DeepLab-v2 network trained on synthetic data only, achieves a mIoU of 37.8% on real data. Then, we analyze the remaining modules by removing one component at a time (row 2

$\mathcal{L}_{G,0}$	$\mathcal{L}^s_{G,1}$	$\mathcal{L}^t_{G,1}$	$\mathcal{L}^t_{G,2}$	$\mathcal{L}_{G,3}$	$\mathcal{T}_f$	mIoU
✓						37.8
✓		✓	✓	✓	✓	39.9
✓	✓		✓	✓	✓	40.3
✓	✓	✓		✓	✓	40.7
✓	✓	✓	✓			41.1
✓	✓	✓	✓	✓		40.6
✓	✓	✓	✓	✓	fix 0.2	40.9
✓	✓	✓	✓	✓	✓	**41.9**

Table 8.8: Ablation results on the Mapillary validation set adapting from GTA5.

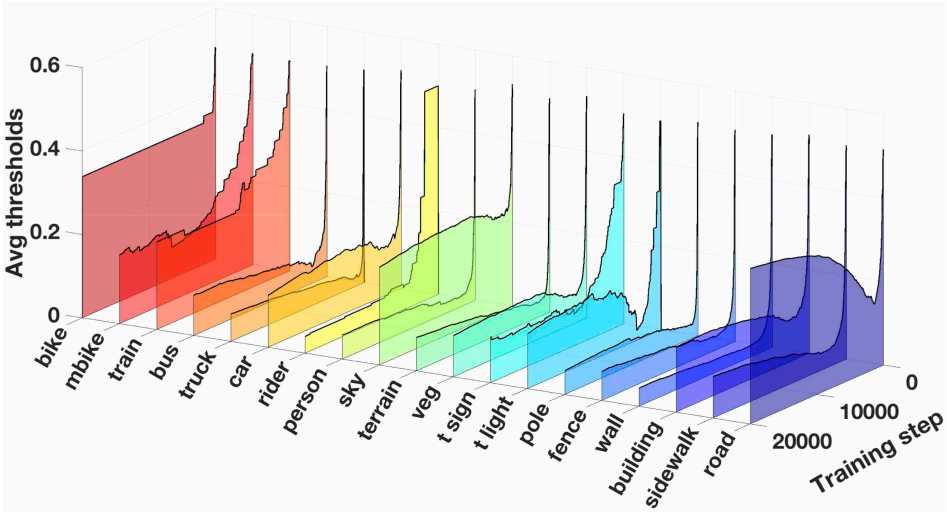


Figure 8.5: Time average over the initial to current step interval of per-class confidence thresholds for different classes and at different training steps on the Mapillary dataset when adapting from GTA5.

to 6). We can appreciate that all the components bring a significant contribution to the final mIoU, which in the full version of the approach where all of them are enabled is 41.9%. The impact of $\mathcal{L}^s_{G,1}$, $\mathcal{L}^t_{G,1}$ and $\mathcal{L}_{G,2}$ is clear by looking at Table 8.8: without each of them the accuracy decreases with respect to the complete framework but remains higher than the source supervised case. In particular, we performed a more detailed analysis of the self-training module. Having no self-training leads to 41.1% of mIoU, while having self-training done on all pixels (without thresholding with $\mathcal{T}_f$) or using a fixed confidence threshold (*e.g.*, setting a constant value of $\mathcal{T}_f = 0.2$ as in [26]) leads to 40.6% and 40.9%, respectively. These results show that self-training is not effective if the reliable pixels are not accurately selected, *e.g.*, if performed immutable over the classes and the training steps. On the other side we found that self-training is effective with confidence thresholds variable over classes and over training time.

We also analyzed the behavior of the per-class time-varying confidence values: they are shown in Figure 8.5 for the GTA5 to Mapillary scenario. Here, we can appreciate how the confidence thresholds vary over the training time and typically converge to a value which may be significantly different among the various classes. While previous works [26, 27] fixed the threshold to 0.2, here we can see how the desired value is variable and ranges between 0.04 and 0.4. Highly confident classes typically have high confidence threshold, in order to propagate back very reliable predictions through self-training as for instance *road*, *sky* and *building*. However, notice that *train* and *bike* have high confidence threshold values only because they are predicted too rarely to be useful for confidence assessment (*i.e.*, the thresholds are piece-wise constant functions), hence represent failure modes of the estimate. More challenging classes such as *wall* and *fence*, instead, are characterized by a much lower confidence value. The most important aspect, however, is to verify that our framework can adapt both to the properties of different classes and to the changes in the network behavior during the training procedure. Additionally, we can compute the mean threshold value averaged over all the classes at the end of the training phase and compare it with previous works [26, 27]. The mean is 0.13 in the considered scenario and 0.19 in the adaptation from GTA5 to Cityscapes, consistently with the value of 0.20 used in previous works [26, 27].

8.4 Summary

In this section, a couple of novel schemes to perform unsupervised domain adaptation from synthetic urban scenes to real-world ones have been proposed under the same common framework. In the first contribution, two different strategies have been used to exploit unlabeled data. The first is based on an adversarial learning framework, based on fully convolutional discriminators. The second is a soft self-teaching strategy, based on the assumption that unsupervised predictions labeled as highly confident by the discriminator are reliable. We also add a region growing module that further refines the confidence maps on the basis of the segmentation output on real-world images. In the second contribution, our framework was enriched by class-aware and time-varying confidence thresholds to adapt to different classes and to different stages of learning of the semantic segmentation network.

Experimental results on the Cityscapes and Mapillary datasets prove the effectiveness of the proposed approach. In particular, we obtained good results both on large-size regions, thanks to the region growing procedure, and on particularly challenging small and uncommon ones, thanks to the class frequency weighting of the self-teaching loss.

Further research will address the exploitation of generative models to produce more realistic and refined synthetic training data to be fed to the framework and the alignment of the latent level representations learned from the two data domain distributions: this is the main focus of the next chapter.

9 Input- and Feature- Level Domain Adaptation

9.1 Introduction

In the previous chapter we have seen how output-level domain adaptation can be performed to leverage the results of an initial stage of training on a source domain distribution. This chapter, instead, explores novel feature-level UDA strategies.

In Section 9.2 we present the first line of research, where we combine an input-level adaptation strategy based on CycleGAN with a naïve feature-level adaptation by means of a couple of discriminator networks [30]. In Section 9.3 we focus more deeply on sole latent-level adaptation by means of features disentanglement via clustering, orthogonality and sparsity [31]. In Section 9.4 we extend previous considerations on feature-level adaptation and we propose a novel metric to quantify the performance of UDA strategies [32, 33].

9.1.1 Contributions

9.2 Input-Level Cyclic Consistency and Feature-Level Adversarial Learning

In a popular work, Hoffman *et al.* [275] introduce the CyCADA framework, which relies on the CycleGAN [29] image-to-image translator for pixel-level adaptation. First of all, a generative module, which is augmented with a semantic consistency constraint to avoid semantic alterations of input data, is trained to perform cross-domain image translation. Then, the generator responsible for the source-to-target mapping is applied on source images before they are provided to the predictor in a supervised training phase. Finally, a separate adversarial feature adaptation step is proposed. Unfortunately, the authors were not able to train the full framework end-to-end with all the proposed modules, due to hardware constraints hindering the insertion of a fully convolutional predictor (*i.e.*, the segmentation network for semantic consistency) inside an already memory-demanding generative module comprising four neural networks. Similarly to [275], Chen *et al.* [276] propose an extension of the CycleGAN framework by introducing a pair of feature domain discriminators and a couple of semantic segmentation networks.

In this section, we introduce an efficient UDA approach combining two strategies: we start performing an image-level (*i.e.*, input-level) domain adaptation using a model based on the Cycle-GAN framework that converts the input synthetic images to the target (real) domain while preserving the semantic content. Then, the data is sent to a MobileNet-v2 architecture [360]

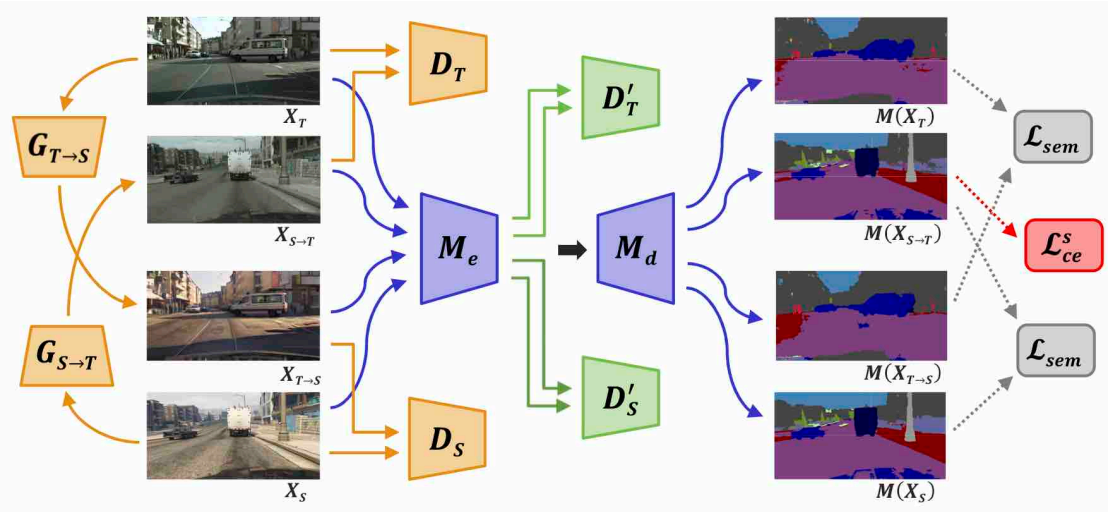


Figure 9.1: Architecture of the proposed framework. Yellow blocks correspond to the CycleGAN framework for image-to-image translation. Original and translated scenes from both source and target sets are projected by the encoder to a latent space on which we apply an extra couple of domain discriminators (green blocks). Structural consistency on generated samples is enforced by the cycle-consistency constraint, whereas semantic uniformity throughout image mapping is promoted by the semantic loss. The segmentation network is reported in blue.

that performs the semantic segmentation. In order to enhance the feature-level adaptation, an additional couple of discriminators working at the intermediate feature level of the network is employed. Moreover, an additional loss component forces also the consistency between the semantic maps, thus avoiding the risk that the domain translation affects the semantic content. Differently from other competing approaches [275, 276] that train independently the various sub-components, we train the complete architecture end-to-end on both synthetic labeled data and unlabeled real-world data in a single optimization framework based on adversarial learning. Finally, using the simple and fast MobileNet-v2 architecture, the inference stage of the approach is suitable for real-time applications as the autonomous driving scenario chosen for the experimental evaluation.

9.2.1 Proposed Approach

Our target is to train a semantic segmentation network in a supervised way on synthetic data and then to adapt it in an unsupervised way to real-world data. We assume to have access to synthetic (*i.e.*, source) labeled images $(x_S, y_S) \in X_S \times Y_S$, as well as to real (*i.e.*, target) unlabeled images $x_T \in X_T$. Due to dissimilar marginal and joint distributions over input and label spaces on both domains, deep models trained on source data struggle to generalize learned knowledge to the target space. To address the effect of domain discrepancy, we resort to a generative approach. We employ an adversarial framework to learn an image-level mapping between source and target spaces. The objective is to produce adapted source images that resemble target ones, while preserving the ground-truth information at our disposal. In this way, we can introduce a form of target supervision by exploiting target-like annotated source images to train the segmentation network.

Figure 9.1 shows the architecture of the proposed framework. As initial step, we adopt a generative approach to learn an image-level mapping for cross-domain image projection. This is

achieved using an adversarial learning scheme exploiting a pair of generator-discriminator couples. Meanwhile, a semantic segmentation network is included to enforce semantic consistency to the generative process. The objective is to perform realistic sample translations, while preserving the semantic structure as identified by the semantic classifier. An additional feature-level adaptation is further included, in the form of a pair of feature discriminators. The way they operate is analogous to their image-level counterparts, as they enforce a statistical alignment of source and target data representations. The key difference lies in the operating space, since they act over an intermediate feature representation produced by the segmentation network, rather than directly within the original image domain.

Our approach is independent of the segmentation architecture and in general any semantic segmentation network can be used, however in our experiments we used the MobileNet-v2 network [360, 361] embedded inside the DeepLab-v3+ framework [4]. The primary component of this widely utilized model is the depthwise separable convolution, a lightweight reinterpretation of the standard convolutional layer responsible for both the efficiency and the reduced weight of the architecture. In addition, inverted residual blocks in place of the standard residual connections further enhance model compactness and size shrinkage.

9.2.1.1 Cycle Consistent Domain Adaptation

The generative module is based on the CycleGAN framework [29]. The source to target (direct) mapping $G_{S\to T}\colon X_S \to X_T$ and the target to source one (inverse mapping) $G_{T\to S}\colon X_T \to X_S$ are discovered by means of an adversarial competition exploiting a couple of discriminators D_S and D_T. The role of the domain discriminators, following the original concept of GANs [108], is to discriminate between *real* images in their original form and *fake* images, *i.e.*, synthetic data subjected to the domain translation. We resort to the standard adversarial objectives at the image level to train separately each generator-discriminator couple:

$$\begin{aligned}&\mathcal{L}_{i,T}\left(G_{S\to T}, D_T, X_S, X_T\right)\\ &\quad= \mathbb{E}_{x_T\sim X_T}\left[\log\left(D_T(x_T)\right)\right]\\ &\quad+ \mathbb{E}_{x_S\sim X_S}\left[\log(1 - D_T(G_{S\to T}(x_S)))\right]\end{aligned} \tag{9.1}$$

$$\begin{aligned}&\mathcal{L}_{i,S}\left(G_{T\to S}, D_S, X_T, X_S\right)\\ &\quad= \mathbb{E}_{x_S\sim X_S}\left[\log\left(D_S(x_S)\right)\right]\\ &\quad+ \mathbb{E}_{x_T\sim X_T}\left[\log(1 - D_S(G_{T\to S}(x_T)))\right]\end{aligned} \tag{9.2}$$

With the aforementioned generative process, we are able to learn a joint source-target distribution starting from the marginal ones, which means we can reproduce the same images in both source and target styles. Unfortunately, since there are infinite joint distributions that match the available marginal ones, we are not guaranteed that the mapping functions we discover are preserving content structure and semantics. In other words, without any additional constraint the $G_{S\to T}$ projection could completely disrupt input source images, still producing new samples with target properties, but far from their original versions. For this reason, we employ an additional loss term enforcing cycle-consistency:

$$\begin{aligned}&\mathcal{L}_{cycle}\left(G_{S\to T}, G_{T\to S}, X_S, X_T\right)\\ &\quad= \mathbb{E}_{x_S\sim X_S}\left[\left\|G_{T\to S}\left(G_{S\to T}\left(x_S\right)\right) - x_S\right\|_1\right]\\ &\quad+ \mathbb{E}_{x_T\sim X_T}\left[\left\|G_{S\to T}\left(G_{T\to S}\left(x_T\right)\right) - x_T\right\|_1\right]\end{aligned} \tag{9.3}$$

The reconstruction requirement provided by the cycle-consistency loss $\mathcal{L}_{cycle}$ should encourage

the preservation of structural properties throughout translations, resulting in a realistic image generation that does not affect the semantic content.

The adversarial strategy we adopt for conditional image generation has proven to be suitable for color and texture changes, but not for more drastic geometrical transformations [108]. This is positive for our target, since we are looking for a cross-domain projection that allows us to safely transfer ground truth information from an original image to its translated version.

9.2.1.2 Enforcing the Semantic Segmentation Consistency across Domains

Following the the idea introduced in [275], we embed the generative module in a task-specific domain adaptation framework. The adversarial architecture is followed by a network performing semantic segmentation on data from both source and target domains. In our work, we will assume the usage of a fully convolutional network, that in our implementation is the MobileNet-v2 network. We will denote it with $M = [M_e, M_d]$, where M_e is the encoder part of the network, while M_d is the decoder. M is pre-trained on the source domain, before its application in the proposed framework.

Due to the lack of labeling data on the target domain, we can not perform supervised training on this domain. However, we introduce a further loss component to enforce the semantic consistency on the generative action: the segmentation network M is supplied with both original and adapted versions of the same image and we measure the semantic discrepancies (*i.e.*, the differences in the segmentation network output) introduced by the projection between domains. The error information is then propagated back through the classifier up to the generator, which is optimized in order to minimize semantic alteration (among other objectives). We impose this semantic uniformity by means of a semantic consistency loss:

$$\begin{aligned} &\mathcal{L}_{sem}\left(G_{S\to T}, G_{T\to S}, M, X_S, X_T\right) \\ &\quad = \mathcal{L}_{ce}\left(M, G_{S\to T}(X_S), \rho(M(X_S))\right) \\ &\quad + \mathcal{L}_{ce}\left(M, G_{T\to S}(X_T), \rho(M(X_T))\right) \end{aligned} \tag{9.4}$$

where $\rho(M(X))$ is the arg max of the output of the semantic segmentation network and:

$$\mathcal{L}_{ce}\left(M, X, Y\right) = -\mathbb{E}_{(x,y)\sim(X,Y)} \sum_{h,w,c} y^{(h,w,c)} \cdot \log(M(x)^{(h,w,c)}) \tag{9.5}$$

Notice that the cross-entropy loss $\mathcal{L}_{ce}$ is computed over segmentation maps obtained by applying the arg max function ρ on semantic predictions $M(X)$, rather than over ground-truth labels. Therefore, its effect is not to promote correct semantic predictions, but to force the generator to yield transformed images that are semantically identical to the original ones when viewed under the scope of M.

This scheme allows us to perform a measure of semantic distance in both domains, without resorting to ground-truth information. On the other side, M is not a perfect predictor (and on the target domain has typically lower performance), therefore an excessive emphasis on this loss may cause undesired artifacts in the generative process.

9.2.1.3 Feature-level Domain Adaptation

Aiming at further improving our domain adaptation framework, we introduce an additional adversarial module to perform feature-level domain adaptation. The core idea is to replicate the adversarial strategy adopted for the image-to-image translation task, where the goal was a pixel-level distribution alignment. The new objective is instead the adaptation of intermediate

feature representations, *i.e.*, to ensure the proper adaptation at the level of the output of the encoder network M_e. The goal now is to make generated images from one domain appear statistically identical to original images from the other domain when looking at their projections in a latent space spanned by the segmentation network. More specifically, we add a couple of feature discriminators and feed them with activations from the output of the MobileNet's feature extractor M_e. The adversarial game, then, takes place between a couple of feature discriminators and the joint action of the two original generators together with the encoder of the segmentation network. The feature-level adversarial objectives are the following:

$$\begin{aligned}&\mathcal{L}_{f,T}(M_e \circ G_{S\to T}, D'_T)\\ &\quad = \mathbb{E}_{x_T\sim X_T}[\log(D'_T(M_e(x_T)))]\\ &\quad + \mathbb{E}_{x_S\sim X_S}[\log(1 - D'_T(M_e(G_{S\to T}(x_S))))]\end{aligned} \tag{9.6}$$

$$\begin{aligned}&\mathcal{L}_{f,S}(M_e \circ G_{T\to S}, D'_S)\\ &\quad = \mathbb{E}_{x_S\sim X_S}[\log(D'_S(M_e(x_S)))]\\ &\quad + \mathbb{E}_{x_T\sim X_T}[\log(1 - D'_S(M_e(G_{T\to S}(x_T))))]\end{aligned} \tag{9.7}$$

Where D'_S and D'_T are the feature discriminators working on data from the source and target domain respectively.

Combining together all the different losses, the full objective becomes:

$$\begin{aligned}\mathcal{L}_{tot} = (\mathcal{L}_{i,S} + \mathcal{L}_{i,T}) + \lambda_{feat}\cdot(\mathcal{L}_{f,S} + \mathcal{L}_{f,T})\\ +\lambda_{cycle}\cdot\mathcal{L}_{cycle} + \lambda_{sem}\cdot\mathcal{L}_{sem} + \lambda_{ce}\cdot\mathcal{L}^s_{ce}\end{aligned} \tag{9.8}$$

We also denote $\mathcal{L}^s_{ce} = \mathcal{L}_{ce}(M, G_{S\to T}(X_S), Y_S)$ the standard cross-entropy loss used to train supervisedly M on source adapted data. The framework optimization then can be expressed as a *min-max* problem:

$$\min_{G_{S\to T}, G_{T\to S}, M}\ \max_{D_S, D_T, D'_S, D'_T} \mathcal{L}_{tot}. \tag{9.9}$$

As a result, we have access to a pair of image-to-image mappings capable of translating images across domains, while, at the same time, making generated samples statistically indistinguishable from true ones when projected into the feature space defined by the segmentation network. Additionally, the semantic segmentation network M is adapted to work on the target data in an unsupervised way (without using target ground truth) thanks to the loss $\mathcal{L}^s_{ce}$. Since all the components are simultaneously trained, when improving the image-to-image mappings also the adaptation of M improves. For a more stable training and to avoid saturation effects [29], the logarithm within adversarial losses is replaced by a L2-norm operator and the objectives are split into separate terms for generators and discriminators individual optimization.

9.2.2 Implementation and Training Details

Network implementation. The image-level generators and discriminators (*i.e.*, $G_{S\to T}$, $G_{T\to S}$, D_S and D_T) are based on the network architectures introduced in [29]. The generators are composed of stride-2 convolutions, residual blocks and fractionally strided convolutions to recover input dimensionality. Discriminators are fully convolutional networks as well, made by the cascade of 5 stride-2 convolutional layers. To avoid excessive size compression, we employ the same structure for feature-level discriminators (D^f_S and D^f_T) as well, but we modify the stride value to 1 for all layers. As concerns the segmentation network M, we employ the DeepLab-v3+ [4] with the MobileNet-v2 [360] as backbone. We select an output stride of 16 for the feature

extractor, whereas for the ASPP block we choose atrous rates of 6, 12 and 18 as suggested by [4].

Training details. Differently from [275], we train our framework in a single shot, so that all the networks are simultaneously optimized according to $\mathcal{L}_{tot}$. We initialize the weights of the semantic predictor from its pre-trained version on the Pascal VOC dataset [114, 362]. We then fine-tune it on the source domain for 90K steps using the standard cross entropy loss before the actual optimization of the adaptation framework. All the other networks are instead trained from scratch. We use the Adam optimizer [363] to train all the components of the proposed approach.

After the initialization of the segmentation network, we train our model for a total of 80K iterations with a single NVDIA GeForce GTX 1080 Ti. The segmentation network is kept fixed for the first 20K steps, until the generators start performing acceptable translations, then we train all the various components together. The large amount of model parameters and the high resolution images from source and target sets prevents us from using full size data for training purposes. To overcome this issue, we extract random patches of 600×600 pixels from training samples, which were previously resized to have a predefined width and the original aspect ratio, and we use them as model inputs.

Concerning the parameters, we experimentally set the terms balancing the various components in $\mathcal{L}_{tot}$ to $\lambda_{cycle} = 20$, $\lambda_{sem} = 0.1$, $\lambda_{feat} = 10^{-4}$ and $\lambda_{ce} = 1$. For the training of the segmentation network we set β_1 (the exponential decay rate for the first moment estimates) to 0.9, and the weight decay to 4×10^{-5}. The learning rate is subject to a polynomial decay of power 0.9, and is decreased to 0 from its initial value (respectively 1×10^{-5} and 5×10^{-6} for the adaptation from the GTA and SYNTHIA datasets). Moreover, we set a batch size of 5 when training the semantic predictor alone in the initial stage, while for full framework optimization, we reduce the batch size to 1 due to memory constraints. For the optimization of the image and feature adversarial models we use a small β_1 term of 0.5 as in [29]. This makes the training process more unstable, but we notice no improvements by changing it. The learning rate for these modules is set to 2×10^{-4}.

We developed our approach in TensorFlow [125] and the code is publicly available at `https://lttm.dei.unipd.it/paper_data/UDAmobile/`.

9.2.3 Experimental Results

In this section, we show some examples of the image translation module. Then, we show the results of the main task, *i.e.*, semantic segmentation on real data, starting from two different synthetic datasets. In order to conclude, some ablation results are presented.

9.2.3.1 Image Domain Translation

The first module in our framework is the image-to-image adaptation in the image space between the synthetic datasets and the real one (Cityscapes) and viceversa. We visualize such cyclic translation in Figure 9.2, where we report the original, adapted and reconstructed images in output from our model for each of the four different considered scenarios. We show both the results when starting from synthetic data and when starting from real scenes.

The most obvious and noticeable difference lies in the shift of colors between real and synthetic images. Synthetic imagery, indeed, are characterized by more vivid colors than the real counterparts hence the adaptation framework needs to compensate for this issue as can be verified in all the proposed qualitative results. Additionally, the textures of some regions (such as the road or the sky) of the images are completely different between the considered domains. In

GTA5 and especially in SYNTHIA (where the road texture is not realistic at all) the road is less uniform than the one present in the Cityscapes dataset, so we could observe that our adaptation framework compensate this aspect. In particular, the model adds some textures on the road when converting an image to the synthetic dataset and it removes such textures when dealing with the opposite task. The sky, instead, tends to appear more blue in the synthetic imagery rather than in the real ones, where it appears more grayish.

Beside those changes in the appearance of the images, we could notice that in general the semantic content and the geometrical structure is preserved unaltered. The objects do not disappear and they do not change position, as we expect. This is ensured by a combination of factors such as the semantic loss, the cycle-consistency loss and the adaptation loss at the feature level which aims at preserving the extraction of similar features from the same objects even if they belong to two different image-spaces.

Furthermore, when moving from synthetic to real domain, in certain pictures we can observe that our model tends to add an ornament to the bottom of the images (*e.g.*, in the second row from GTA5 to Cityscapes). Although this is irrelevant for the semantic segmentation task, since we exclude the car on which the camera is mounted, we argue that the image-space adaptation is leading to reasonable outcomes because it tries to replicate the trademark of the car used for the Cityscapes data acquisition.

A few artifacts are present especially in the sky region where the model tends to add some shadows coming from clouds or buildings present in other images: this can be noticed in the first and third adapted images from the GTA5 dataset. However, it is noticeable that the cycle-consistency helps the model to recover the exact appearance of the sky of such images.

Table 9.1: mIoU on the different classes of the Cityscapes validation set. The approaches have been trained in a supervised way on the GTA5 dataset and the unsupervised domain adaptation has been performed using the Cityscapes training set. The mean and per class highest results have been highlighted in bold.

	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mean
Source only	23.1	13.1	42.6	2.3	13.9	5.0	10.3	8.0	68.6	6.7	24.5	40.8	0.3	48.1	9.4	16.3	0.0	0.0	0.0	17.5
Ours (full)	**87.6**	**36.7**	**83.5**	**29.1**	**17.8**	**33.6**	24.3	**35.2**	**83.1**	**28.9**	**76.3**	**59.1**	**14.0**	**85.9**	**25.4**	**29.4**	**2.6**	**19.5**	**9.3**	**41.1**
CycleGAN [29]	84.9	36.4	74.3	12.9	7.1	23.6	7.9	19.9	60.2	13.4	45.8	46.8	5.4	72.4	18.0	22.3	0.8	3.2	0.5	29.3
CyCADA [275]	83.8	35.3	80.4	20.7	15.7	28.4	**27.0**	24.8	80.2	23.1	69.0	56.6	11.5	80.8	23.4	27.0	2.4	12.4	5.2	37.3
Target	97.7	81.9	91.0	47.6	50.1	58.4	62.3	73.4	91.4	59.8	94.3	77.2	50.5	93.2	59.2	74.8	55.8	49.5	73.0	70.6

9.2.3.2 Adaptation from the GTA5 Dataset

The first set of experiments regards the unsupervised adaptation of the semantic segmentation network M to the Cityscapes dataset after an initial stage of supervised training on the GTA5 dataset. To evaluate the adaptation performance of our framework we computed the mean Intersection over Union (mIoU) between model predictions and the relative ground truth label maps for the scenes in the Cityscapes validation set.

The results of these synthetic to real adaptation experiments are summarized in Table 9.1. The baseline approach, *i.e.*, the supervised training of the semantic segmentation on the GTA5 (source) dataset followed by testing on Cityscapes without any adaptation, leads to a very low mIoU of 17.5% (first row). When compared to the training of the same network on the target (Cityscapes) dataset (last row, denoted as *Target*), the huge difference reveals the struggle of the predictor to overcome the statistical discrepancy between the source and target domains.

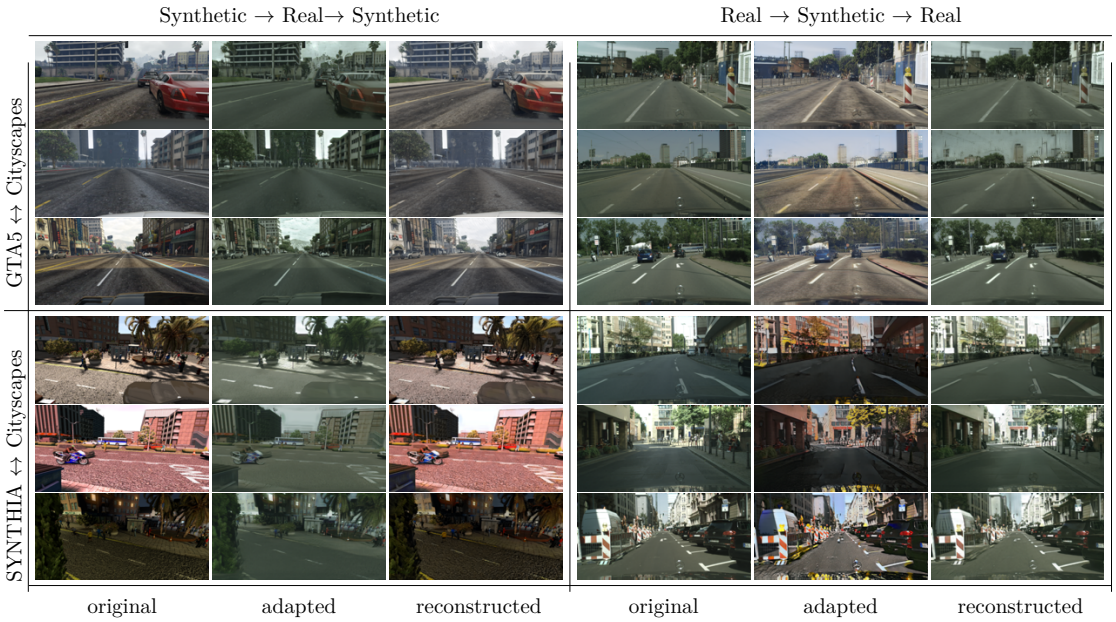


Figure 9.2: Examples of image translations (adaptation and reconstruction of the original image) in the four different cases considered. In the first quadrant (up-left of 3×3 images) we move from the GTA5 dataset to the adapted images in the domain of the Cityscapes dataset to the reconstructed images in the GTA5 domain. The second quadrant (up-right) shows respectively: the starting Cityscapes images, their translation to the GTA dataset and the reconstructed images in the domain of the Cityscapes images. The third (down-left) and the fourth (down-right) quadrant are analogous to the first and the second ones using the SYNTHIA dataset in place of GTA5.

Our unsupervised domain adaptation method brings a huge improvement over the naïve source only approach, reaching a mIoU of 41.1% when employed in its full extent (5th row), with a performance boost of 23.6% over the baseline. Moreover, the accuracy enhancement is well distributed over all the semantic classes, from the most common ones (*e.g. road*, *building*, *sky*), to the less frequent categories (*e.g. train*, *motobike*, *bike*). In particular notice that some of the less frequent are never recognized when no adaptation is performed while the proposed approach allows to detect them even if they remain very challenging. This proves the effectiveness of our model in mitigating the statistical discrepancy through a combined feature and pixel level alignment.

We compared the results we obtained on the Cityscapes validation set with two approaches of the same family: namely, CycleGAN [29] and CyCADA [275]. Those are very well known approaches and also represent the starting point for some architectural design choices we made. For comparison purposes, we used the CycleGAN implementation from [364] and we implemented the framework of [275] from scratch and inserted in it the same segmentation network we used in our tests (*i.e.*, DeepLabV3+ [4] with MobileNet-v2 [360] as backbone). As regards the training details of the compared methodologies, we employed the standard training procedures proposed in the respective papers: for CycleGAN [29] we train the model with rescaled images, while for CyCADA [275] we train the model extracting random patches after a rescaling operation.

From Tables 9.1 and 9.2 we can appreciate that our method is able to outperform CyCADA by about 4% and 5% respectively, which is uniformly distributed among classes: notice how our approach is the best on 18 out of 19 classes (it is outperformed only by [275] on the traffic

light class). This has to be mostly attributed to the feature alignment process which directly influences the generative action and also to the simultaneous optimization of all framework components. Additionally, we could observe that CycleGAN has a much lower accuracy and lies in between the training made only on source data and our full method. Indeed, it consists of a simpler framework where only the image translation based on the cycle-consistency constraint is present.

Figure 9.3 displays some qualitative results in terms of semantic prediction maps for different adaptation strategies. The first and second columns include the original Cityscapes RGB images and their corresponding label maps, while the last 3 columns show the segmentation outputs with no adaptation (*i.e.*, *source only*), using the approach of [275] and with the proposed unsupervised domain adaptation strategy.

The first 4 rows of Figure 9.3 show the remarkable improvement achieved over the *source only* baseline when resorting to the GTA5 dataset as source domain. The effect of the adaptation is to boost the detection capability of the predictor M, as it manages to obtain a more accurate understanding of the input target scene, both in terms of correct categorization and precise spatial identification of the different semantic entities. All semantic classes happen to highly benefit from the adaptation, leading to generally better semantic predictions. Anyway some categories exhibit more noticeable improvements, such as the *road* and *sky* classes as it is possible to see on the predicted maps in rows 1, 2 and 4.
Column 4 of Figure 9.2 shows the output of [275]: our method outperforms [275] on the majority of the semantic classes, with a particularly noticeable improvement on some of them. For instance, our approach in general leads to a more precise segmentation of the upper portion of target scenes, mainly involving the *sky* and *building* classes. It is possible to notice how the approach of [275] sometimes confuses part of *sky* as *terrain* or *building* (rows 1, 2 and 4). As previously stated, this is due to the dataset bias occurring between source and target domains in terms of color shift and change of texture, making the detection of semantic components quite hard to achieve, especially when the classification involves different categories (such as *sky* and *building*) sharing a common appearance on the source and target domains. Furthermore, we observe that less frequent classes corresponding to small image details (*e.g.*, *traffic sign*) are more precisely localized, as well as more common categories lacking an always definite semantic categorization (*e.g. wall* and *sidewalk*). Once again, this remarks the capability of our adaptation framework to address domain discrepancy with the combined pixel and feature level alignment.

9.2.3.3 Adaptation from the SYNTHIA Dataset

In the second set of experiments we changed the source domain, replacing the GTA5 dataset with the SYNTHIA one. Table 9.2 shows the numerical results we got from the evaluation. As before, we started by training the semantic segmentation network on synthetic data and measuring its performance on the Cityscapes validation set, which we employ as a baseline (first row). The accuracy (18.2%) happens to be slightly higher than in the first scenario, even if the adaptation task is more challenging due to a wider dataset bias. Anyway, the huge performance gap with respect to the target-based supervised optimization still persists, leaving large room for improvement.

Our adaptation framework managed to reduce the domain discrepancy quite successfully boosting the mIoU up to 32.6% from the original 18.2% of the baseline, with an improvement of almost 15%. As for the GTA5 to Cityscapes adaptation, the accuracy for all semantic categories benefit from the unsupervised adaptation. For example, road segmentation is strongly improved, starting from a mIoU for the road class of 1.4% (denoting basically no detection capability) and reaching a final accuracy of 53.8%. This class is particularly interesting since the reason for the

Table 9.2: mIoU on the different classes of the Cityscapes validation set. The approaches have been trained in a supervised way on the SYNTHIA dataset and the unsupervised domain adaptation has been performed using the Cityscapes training set. The mean and per class highest results have been highlighted in bold.

	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	sky	person	rider	car	bus	mbike	bike	mean
Source only	1.4	10.6	29.1	1.0	0.0	17.2	2.0	3.6	68.5	65.0	42.3	0.1	41.7	8.2	0.0	0.5	18.2
Ours (full)	**53.8**	21.3	**69.4**	3.7	**0.1**	**31.6**	3.5	**12.6**	**77.5**	**75.2**	**51.9**	**13.2**	**64.1**	**15.9**	**10.8**	**16.7**	**32.6**
CycleGAN [29]	42.9	**26.7**	44.6	0.5	**0.1**	29.2	3.5	5.8	67.1	70.8	46.0	3.4	32.8	7.0	2.4	3.7	24.2
CyCADA [275]	30.3	15.8	64.0	**5.9**	0.0	30.6	**3.8**	10.4	76.4	73.0	42.9	4.9	54.3	15.0	3.0	8.9	27.5
Target	97.8	83.7	91.2	47.7	49.7	58.8	63.0	73.9	92.4	94.4	77.9	53.7	94.1	80.5	44.7	73.5	73.6

low accuracy lies in the rather unrealistic synthetic road texture of SYNTHIA images, whose semantic attributes are not easily generalizable to the Cityscapes dataset. To that end, our framework introduces a substantial distribution alignment both at pixel level and inside the intermediate feature space. We once again compared the adaptation performance of our model with the one of CyCADA [275] in the third row. Our framework is more effective than the competitor, with an average mIoU increase of 5%, shared by the majority of the classes. For instance, the road accuracy is significantly enhanced by our model, proving the better capability of reducing domain discrepancy. Again, the CycleGAN [29] approach has intermediate results between the source only approach and CyCADA [275].

Some qualitative results for the adaptation from the SYNTHIA dataset are shown last 4 rows of Figure 9.3. They confirm the numerical evaluation: the strong diversity between the Cityscapes and SYNTHIA datasets negatively affects the semantic understanding of the predictor, as clearly visible in the third column showing the outputs of the baseline approach with no adaptation. Even common classes such as *road* and *building* suffer from quite flawed semantic detection leading to almost no understanding of target scene structure, with a worse performance when compared to the GTA5 scenario. This has to be ascribed to the poor realism of synthetic images and to the variable view point of SYNTHIA scenes, which are captured from multiple camera angles and not exclusively from a car perspective as for the Cityscapes images. The adaptation successfully mitigates the domain discrepancy providing the predictor with an enhanced perception of the semantic morphology of target inputs. For example, the *road*, which without adaption is incorrectly classified as *building*, probably due to its quite unrealistic texture on source images, after the adaptation is detected with a much greater accuracy.

Furthermore, our approach shows some improvement also with respect to the method proposed in [275]. For example, the adaption following [275] struggles in the correct segmentation of *road* and *sidewalk* classes, which are easily mistaken one for the other, an issue greatly reduced in the maps of our approach in the last column. At the same time, semantic predictions exhibit a better detection accuracy on objects belonging to low frequency classes, such as *motorbike* and *bike*, highlighting an increased robustness of our method due to the combined pixel and feature level alignment and to the simultaneous optimization of all the network components.

9.2.3.4 Ablation Studies

Finally, we analyze in detail the performance gain brought by the different components of our framework. For this evaluation we considered the domain adaptation from GTA5 to Cityscapes. A relevant contribution is due to the CycleGAN-based image-to-image translator, which effectively bridges the domain gap at the pixel level by generating realistic target-like labeled data and pushes the accuracy on the Cityscapes validation set up to 39.1% (second row in Table 9.3).

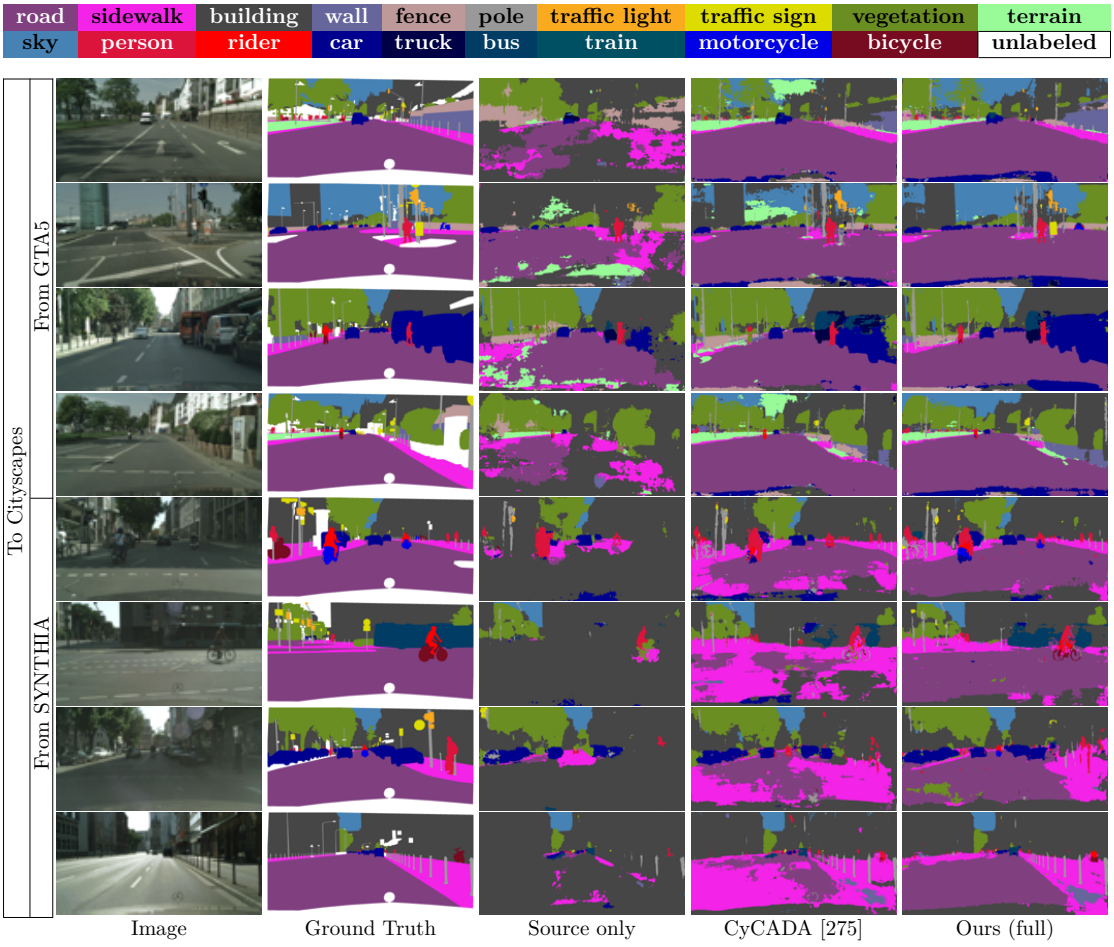


Figure 9.3: Semantic segmentation of some sample scenes extracted from the Cityscapes validation set when adapting source knowledge learned on the GTA5 (rows $1-4$) and SYNTHIA (rows $5-8$) datasets.

However, notice that this value is significantly different from the original CycleGAN [29] result reported in Table 9.1 because in our framework the optimization of the segmentation network is made jointly with the cyclic translation and the training considers patches of images. Then, we evaluated the impact on the final adaptation performance of the semantic and feature-based losses by alternately setting to 0 the λ_{sem} and λ_{feat} parameters. The regularizing action of the feature level adaptation allows to increase the accuracy to 39.6% (third row in Table 9.3) with a small but noticeable impact. The semantic loss (fourth row in Table 9.3) has a larger impact, leading to an improvement of 1.6% on the final score. Finally, by using both components together we obtain a combined improvement of 2%, leading to the final accuracy of 41.1%.

Table 9.3: Ablation study on the Cityscapes validation set. The approaches have been trained in a supervised way on the GTA5 dataset and the unsupervised domain adaptation has been performed using the Cityscapes training set. The mean and per class highest results have been highlighted in bold.

	road	sidewalk	building	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mean
Source only	23.1	13.1	42.6	2.3	13.9	5.0	10.3	8.0	68.6	6.7	24.5	40.8	0.3	48.1	9.4	16.3	0.0	0.0	0.0	17.5
Ours ($\lambda_{sem}, \lambda_{feat} = 0$)	87.8	**39.1**	81.0	24.8	16.0	31.9	27.2	25.7	78.6	22.9	69.7	55.5	**16.7**	85.3	25.7	31.0	**4.4**	14.7	5.6	39.1
Ours ($\lambda_{sem} = 0$)	**88.3**	37.1	81.1	25.4	15.3	**33.6**	**29.5**	28.8	80.0	24.4	69.2	56.0	15.6	85.0	25.5	30.8	3.9	16.3	6.2	39.6
Ours ($\lambda_{feat} = 0$)	87.3	36.6	82.8	**29.1**	**19.9**	32.9	24.9	32.3	82.8	28.1	74.6	58.3	11.6	**85.9**	**26.3**	**31.9**	1.3	**19.5**	6.8	40.7
Ours (full)	87.6	36.7	**83.5**	**29.1**	17.8	**33.6**	24.3	**35.2**	**83.1**	**28.9**	**76.3**	**59.1**	14.0	**85.9**	25.4	29.4	2.6	**19.5**	**9.3**	**41.1**
Target	97.7	81.9	91.0	47.6	50.1	58.4	62.3	73.4	91.4	59.8	94.3	77.2	50.5	93.2	59.2	74.8	55.8	49.5	73.0	70.6

9.3 Feature-Level Regularization

In this contribution we focus more closely to latent-level adaptation alone. We propose to disentangle the latent representations resorting to a feature clustering method that captures the different semantic modes of the feature distribution and groups features of the same class into tight and well-separated clusters. Furthermore, we introduce two novel learning objectives to enhance the discriminative clustering performance: an orthogonality loss forces spaced out individual representations to be orthogonal, while a sparsity loss reduces class-wise the number of active feature channels. The joint effect of these modules is to regularize the structure of the feature space.

As we observed there are three main levels to which adaptation may occur [13]: namely, at the input, features or output stages. A popular solution has become to bridge the domain gap at an intermediate or output representation level by means of adversarial techniques [27,273,275]. The major drawback of these kind of approaches is that they usually perform a semantically unaware alignment, as they neglect the underlying class-conditional data distribution. Additionally, they typically require a long training time to converge and the process may be unstable.

Differently from these techniques, our approach is simple and does not require complex adversarial learning schemes: it entails only a slight increase of computation time with respect to the sole supervised learning. The main idea is depicted in Figure 9.4: we devise a domain adaptation technique specifically targeted to guide the latent space organization and driven by 3 main components. The first is a feature clustering method developed to group together features of the same class, while pushing apart features belonging to different classes. This constraint, which works on both domains, is similar in spirit to the recent progresses in contrastive learning for classification problems [133]; however, it has been developed aiming at a simpler computation, as the number of features per image is significantly larger than in the classification task. The second is a novel orthogonality requirement for the feature space, aiming at reducing the cross-talk

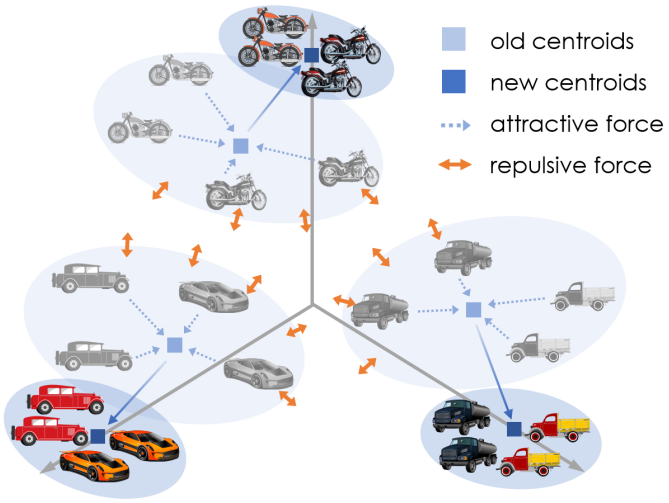


Figure 9.4: The proposed domain adaptation scheme is driven by 3 main components, *i.e.*, feature clustering, orthogonality and sparsity. These push features in the previous step (in light gray) to new locations (colored) where features of the same class are clustered, while features of distinct classes are pushed away. To further improve performance, features of distinct classes are forced to be orthogonal and sparse.

between features belonging to different classes. Finally, a sparsity constraint is added to reduce the number of active channels for each feature vector: our aim is to enforce the capability of deep learning architectures to learn a compact representation of the scene. The combined effect of these modules allows to regularize the structure of the latent space in order to encompass the source and target domains in a shared representation.

Summarizing, our main contributions are: (1) we extend feature clustering (similarly to contrastive learning) to semantic segmentation; (2) we introduce orthogonality and sparsity objectives to force a regular structure of the embedding space; (3) we achieve state-of-the-art results on feature-level adaptation on two widely used benchmarks.

9.3.1 Proposed Approach

In this section, we provide an in-depth description of the core modules of the proposed method. Our approach leverages a clustering objective applied over the individual feature representations, with novel orthogonality and sparsity constraints. Specifically, inter- and intra- class alignments are enforced by grouping together features of the same semantic class, while simultaneously pushing away those of different categories. By enforcing the clustering objective on both source and target representations, we drive the model towards feature-level domain alignment. We further regularize the distribution of latent representations by the joint application of an orthogonality and a sparsity losses. The orthogonality module has a two-fold objective: first, it forces feature vectors of kindred semantic connotations to activate the same channels, while turning off the remaining ones; second, it constrains feature vectors of dissimilar semantic connotations to activate different channels, *i.e.*, with no overlap, to reduce cross interference. The sparsity objective further encourages a lower volume of active feature channels from latent representations, *i.e.*, it concentrates the energy of the features on few dimensions.

A graphical outline of the approach with all its components is shown in Figure 9.5: the training objective is given by the combination of the standard supervised loss with the proposed

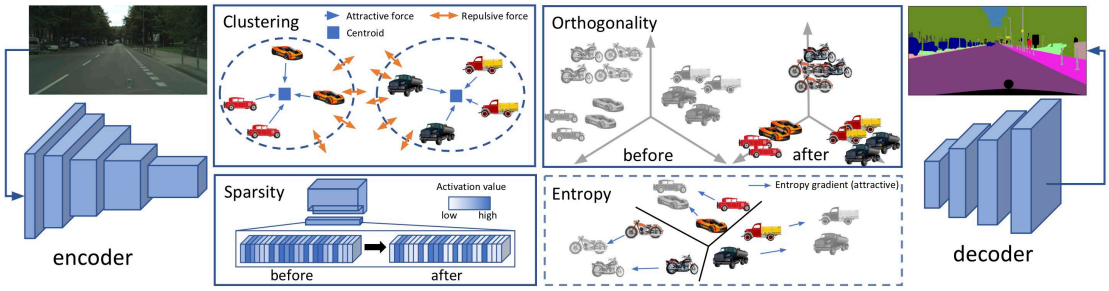


Figure 9.5: Overview of the proposed approach. Features after supervised training on the source domain are represented in light gray, while features of the current step are colored. A set of techniques is employed to better shape the latent feature space spanned by the encoder. Features are clustered and the clusters are forced to be disjoint. At the same time, features belonging to different classes are forced to be orthogonal with respect to each other. Additionally, features are forced to be sparse and an entropy minimization loss could also be added to guide target samples far from the decision boundaries.

adaptation modules, *i.e.*, it is computed as:

$$\mathcal{L}'_{tot} = \mathcal{L}_{ce} + \lambda_{cl} \cdot \mathcal{L}_{cl} + \lambda_{or} \cdot \mathcal{L}_{or} + \lambda_{sp} \cdot \mathcal{L}_{sp} \tag{9.10}$$

where $\mathcal{L}_{ce}$ is the standard supervised cross entropy loss. The other components will be detailed in the following sections: the main clustering objective ($\mathcal{L}_{cl}$) is introduced in Section 9.3.1.1. The orthogonality constraint ($\mathcal{L}_{or}$) is discussed in Section 9.3.1.2 and finally the sparsity constraint ($\mathcal{L}_{sp}$) is detailed in Section 9.3.1.3. The λ parameters balance the multiple losses and are experimentally chosen using a validation set.
In addition, we further integrate the proposed adaptation method with an off-the-shelf entropy-minimization like objective ($\mathcal{L}_{em}$), to provide an extra regularizing action over the segmentation feature space and ultimately achieve an improved performance in some evaluation scenarios. In particular, we adopt the simple, yet effective, maximum squares objective of [325], in its *image-wise class-balanced* version. Hence, we can define the ultimate training objective comprising the entropy module as:

$$\mathcal{L}_{tot} = \mathcal{L}'_{tot} + \lambda_{em} \cdot \mathcal{L}_{em} \tag{9.11}$$

9.3.1.1 Discriminative Clustering

In the considered UDA setting, we are provided with plenty of samples $\mathbf{X}_n^s \in \mathbb{R}^{H \times W \times 3}$ from a source dataset, in conjunction with their semantic maps $\mathbf{Y}_n^s \in \mathbb{R}^{H \times W}$. Those semantic maps contain at each spatial location a ground truth index belonging to the set of possible classes $\mathcal{C}$, which denotes the semantic category of the associated pixel. Concurrently, we have at our disposal target training samples $\mathbf{X}_n^t \in \mathbb{R}^{H \times W \times 3}$ with no label maps (we allow only the availability of a small amount of target labels for validation and testing purposes). Despite sharing similar high-level semantic content, the source and training samples are distributed differently, preventing a source-based model to achieve a satisfying prediction accuracy on target data without adaptation. We denote as $S = F \circ C$ the segmentation network composed of an encoder and a decoder modules, namely the feature extractor F and the classifier C. Notice that the proposed method is agnostic to the employed deep learning model, except for the assumption of an auto-encoder structure and of positive feature values as provided by ReLU activations that are typically placed at the encoder output (as almost all the current state-of-the-art approaches for semantic segmentation).

To bridge the domain gap between the source and target datasets we operate at the feature level. The discrepancy of input statistics across domains is reflected into a shift of feature distribution in the latent space spanned by the feature extractor. This ultimately may cause the source-trained classifier to draw decision boundaries crossing high density regions of the target latent space [151], since it is inherently unaware of the target semantic modes extracted from unlabeled target data. Thus, the classification performance over the target domain is strongly degraded when compared to the upper bound of the source prediction accuracy.

We cope with this performance degradation by resorting to a clustering module, that serves as constraint towards a class-conditional feature alignment between domains. Given a batch of source ($\mathbf{X}_n^s$) and target ($\mathbf{X}_n^t$) training images (for ease of notation we pick a single image per domain), we first extract the feature tensors $\mathbf{F}_n^s = F(\mathbf{X}_n^s)$ and $\mathbf{F}_n^t = F(\mathbf{X}_n^t)$, along with the computed output segmentation maps $\mathbf{S}_n^s = S(\mathbf{X}_n^s)$ and $\mathbf{S}_n^t = S(\mathbf{X}_n^t)$. The clustering loss is then computed as:

$$\mathcal{L}_{cl} = \frac{1}{|\mathbf{F}_n^{s,t}|} \sum_{\substack{\mathbf{f}_i \in \mathbf{F}_n^{s,t} \\ \hat{y}_i \in \mathbf{S}_n^{s,t}}} d(\mathbf{f}_i, \mathbf{c}_{\hat{y}_i}) - \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{j \in \mathcal{C}} \sum_{\substack{k \in \mathcal{C} \\ k \neq j}} d(\mathbf{c}_j, \mathbf{c}_k) \tag{9.12}$$

where $\mathbf{f}_i$ is an individual feature vector corresponding to a single spatial location from either source or target domain and $\hat{y}_i$ is the corresponding predicted class (to compute $\hat{y}_i$ the segmentation map $\mathbf{S}_n^{s,t}$ is downsampled to match the feature tensor spatial dimensions). The function $d(\cdot)$ represents a generic distance measure, that we set to the $L1$ norm (we also tried the $L2$ norm but it yielded lower results). Finally, $\mathbf{c}_j$ denotes the centroid of semantic class $j \in \mathcal{C}$ computed according to the standard formula:

$$\mathbf{c}_j = \frac{\sum_{\mathbf{f}_i} \sum_{\hat{y}_i} \delta_{j,\hat{y}_i} \mathbf{f}_i}{\sum_{\hat{y}_i} \delta_{j,\hat{y}_i}}, \quad j \in \mathcal{C} \tag{9.13}$$

where $\delta_{j,\hat{y}_i}$ is equal to 1 if $\hat{y}_i = j$, and to 0 otherwise.

The clustering objective is composed of two terms, the first measures how close features are from their respective centroids and the second how spaced out clusters corresponding to different semantic classes are. Hence, the effect provided by the loss minimization is twofold: firstly, feature vectors from the same class but different domains are tightened around class feature centroids; secondly, features from separate classes are subject to a repulsive force applied to feature centroids, moving them apart.

9.3.1.2 Orthogonality

As opposed to previous works on clustering-based adaptation methods for image classification, in semantic segmentation additional complexity is brought by the dense structured classification. To this end, we first introduce an orthogonality constraint in the form of a training objective. More precisely, feature vectors from either domains, but of different semantic classes according to the network predictions, are forced to be orthogonal, meaning that their scalar product should be small. On the contrary, features sharing semantic classification should carry high similarity, *i.e.*, large scalar product. Yet, feature tensors associated to training samples enclose thousands of feature vectors to cover the entire spatial extent of the scene and to reach pixel-level classification. Thus, since measuring pair-wise similarities requires a significant computational effort, we calculate the scalar product between each feature vector and every class centroid $\mathbf{c}_j$ (centroids are computed using Eq. (9.13)). Inspired by [244, 250], we devise the orthogonality objective as an entropy minimization loss that forces each feature to be orthogonal with respect

to all the centroids but one:

$$\mathcal{L}_{or} = - \sum_{\mathbf{f}_i \in F(\mathbf{X}_n^{s,t})} \sum_{j \in \mathcal{C}} p_j(\mathbf{f}_i) \log p_j(\mathbf{f}_i) \tag{9.14}$$

where $\{p_j(\mathbf{f}_i)\}$ denotes a probability distribution derived as:

$$p_j(\mathbf{f}_i) = \frac{e^{\langle \mathbf{f}_i, \mathbf{c}_j \rangle}}{\sum_{k \in \mathcal{C}} e^{\langle \mathbf{f}_i, \mathbf{c}_k \rangle}}, \quad j \in \mathcal{C} \tag{9.15}$$

The loss minimization forces a peaked distribution of the probabilities $\{p_j(\mathbf{f}_i)\}$, promoting the orthogonality property as described above, since each feature vector is compelled to carry a high similarity score with a single class centroid. The overall effect of the orthogonality objective is to promote a regularized feature distribution, which should ultimately boost the clustering efficacy in performing domain feature alignment.

9.3.1.3 Sparsity

To strengthen the regularizing effect brought by the orthogonality constraint, we introduce a further training objective to better shape class-wise feature structures inside the latent space. In particular, we propose a sparsity loss, with the intent of decreasing the number of active feature channels of latent vectors. The objective is defined as follows:

$$\mathcal{L}_{sp} = - \sum_{i \in \mathcal{C}} ||\tilde{\mathbf{c}}_i - \boldsymbol{\rho}||_2^2 \tag{9.16}$$

where $\tilde{\mathbf{c}}_i$ stands for the normalized centroid $\mathbf{c}_i$ in $[0,1]^D$ and D denotes the number of feature maps in the encoder output. We also empirically set $\boldsymbol{\rho} = [0.5]^D$. It can be noted that the sparsifying action is delivered on class centroids, thus applying an indirect, yet homogeneous, influence over all feature vectors from the same semantic category. The result is a semantically-consistent suppression of weak activations, while rather active ones are jointly raised.

While the orthogonality objective aims at promoting different sets of activations on feature vectors from separate semantic classes, the sparsity loss seeks to narrow those sets to a limited amount of units. Again, the goal is to ease the clustering loss task in creating tight and well distanced aggregations of features of similar semantic connotation from either source and target domains, by providing an improved regularity to the class-conditional semantic structures inside the feature space.

9.3.2 Experimental Setup

Model Architecture. The modules introduced in this work are agnostic to the underlying network architecture and can be extended to other scenarios. For fair comparison with previous works [151,282,325] we employ the DeepLab-V2, a fully convolutional segmentation network with ResNet-101 [152] or VGG-16 [365] as backbones. Further details on the segmentation network architecture can be found in [151, 282], as we follow the same implementation adopted in those works. We initialize the two encoder networks with ImageNet [121] pretrained weights. In addition, prior to the actual adaptation phase, we supervisedly train the segmentation network on source data.

Training Details. The model is trained with the starting learning rate set to 2.5×10^{-4} and decreased with a polynomial decay rule of power 0.9. We employ weight decay regularization

Table 9.4: Numerical evaluation of the GTA5 and SYNTHIA to Cityscapes adaptation scenarios in terms of per-class and mean IoU. Evaluations are performed on the validation set of the Cityscapes dataset. In all the experiments the DeepLab-V2 segmentation network is employed, with VGG-16 (top) or ResNet-101 (bottom) backbones. The mIoU* results in the last column refer to the 13-classes configuration, *i.e.*, classes marked with * are ignored. MaxSquares IW (r) denotes our re-implementation, as original results are provided only for the ResNet-101 backbone.

	B	Method	Road	Sidewalk	Building	Wall*	Fence*	Pole*	T. Light	T. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU (all)	mIoU* (13-cl)
GTA5 → Cityscapes	VGG16	Source Only	26.5	13.3	45.1	6.0	15.2	16.5	21.3	8.5	78.0	8.3	59.7	45.0	10.5	69.1	22.8	17.9	0.0	16.4	2.7	25.4	-
		FCNs ITW [273]	70.4	**32.4**	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1	-
		CyCADA (feat) [275]	85.6	30.7	74.7	14.4	13.0	17.6	13.7	5.8	74.6	15.8	69.9	38.2	3.5	72.3	16.0	5.0	0.1	3.6	0.0	29.2	-
		CBST [323]	66.7	26.8	73.7	14.8	9.5	**28.3**	25.9	10.1	75.5	15.7	51.6	**47.2**	6.2	71.9	3.7	2.2	**5.4**	**18.9**	**32.4**	30.9	-
		MinEnt [151]	85.1	18.9	76.3	**32.4**	**19.7**	19.9	21.0	8.9	76.3	**26.2**	63.1	42.8	5.9	**80.8**	**20.2**	9.8	0.0	14.8	0.6	32.8	-
		MaxSquare IW (r) [325]	81.4	20.0	75.4	19.4	19.1	16.1	24.4	7.9	78.8	22.9	65.9	45.0	12.3	74.6	16.1	10.3	0.2	11.3	1.0	31.7	-
		Ours ($\mathcal{L}'_{tot}$)	83.6	16.6	79.0	19.8	18.7	21.5	27.3	**15.9**	80.2	14.3	**72.6**	47.0	17.5	76.8	16.6	**13.9**	0.1	16.0	3.4	33.7	-
		Ours ($\mathcal{L}_{tot}$)	**86.0**	13.5	**79.4**	20.4	18.5	21.5	**27.6**	15.2	**80.8**	21.9	**72.6**	46.3	**18.1**	80.0	16.9	13.1	1.0	14.6	2.0	**34.2**	-
	ResNet101	Source Only	81.8	16.3	74.4	18.6	12.7	23.5	29.3	18.1	73.5	21.4	77.6	55.6	25.6	74.1	28.6	10.2	3.0	25.8	32.7	37.0	-
		AdaptSegNet (feat) [282]	83.7	27.6	75.5	20.3	19.9	27.4	28.3	27.4	79.0	28.4	70.1	55.1	20.2	72.9	22.5	35.7	**8.3**	20.6	23.0	39.3	-
		MinEnt [151]	84.4	18.7	80.6	23.8	23.2	28.4	36.9	23.4	83.2	25.2	**79.4**	59.0	**29.9**	78.5	33.7	29.6	1.7	29.9	33.6	42.3	-
		SAPNet [366]	88.4	38.7	79.5	**29.4**	**24.7**	27.3	32.6	20.4	82.2	32.9	73.3	55.5	26.9	82.4	31.8	41.8	2.4	26.5	24.1	43.2	-
		MaxSquare IW [325]	89.3	**40.5**	81.2	29.0	20.4	25.6	34.4	19.0	83.6	34.4	76.5	59.2	27.4	83.8	**38.4**	43.6	7.1	**32.2**	32.5	45.2	-
		Ours ($\mathcal{L}'_{tot}$)	88.7	32.2	81.8	24.1	22.1	**30.8**	**37.6**	**32.8**	83.4	36.3	76.0	60.0	27.0	81.0	34.2	43.0	8.0	23.4	38.1	45.3	-
		Ours ($\mathcal{L}_{tot}$)	**89.4**	30.7	**82.1**	23.0	22.0	29.2	**37.6**	31.7	**83.9**	**37.9**	78.3	**60.7**	27.4	**84.6**	37.6	**44.7**	7.3	26.0	**38.9**	**45.9**	-
SYNTHIA → Cityscapes	VGG16	Source Only	7.8	13.7	66.6	2.2	0.0	23.9	4.8	13.3	71.2	-	76.5	49.2	12.1	67.1	-	24.5	-	9.8	9.2	28.3	32.8
		FCNs ITW [273]	11.5	19.6	30.8	4.4	0.0	20.3	0.1	11.7	42.3	-	68.7	51.2	3.8	54.0	-	3.2	-	0.2	0.6	20.2	22.9
		Cross-City [277]	62.7	25.6	**78.3**	-	-	-	1.2	5.4	81.3	-	81.0	37.4	6.4	63.5	-	16.1	-	1.2	4.6	-	35.7
		CBST [323]	69.6	28.7	69.5	**12.1**	**0.1**	**25.4**	11.9	13.6	**82.0**	-	**81.9**	49.1	14.5	66.0	-	6.6	-	3.7	**32.4**	35.4	36.1
		MinEnt [151]	37.8	18.2	65.8	2.0	0.0	15.5	0.0	0.0	76.0	-	73.9	45.7	11.3	66.6	-	13.3	-	1.5	13.1	27.5	32.5
		MaxSquare IW (r) [325]	9.1	12.7	72.5	1.0	0.0	22.3	7.0	8.4	80.0	-	77.9	49.4	10.0	71.8	-	23.8	-	6.0	13.5	29.1	34.0
		Ours ($\mathcal{L}'_{tot}$)	**78.5**	29.9	77.7	1.2	**0.1**	24.1	11.9	**15.0**	78.7	-	78.5	51.0	15.4	73.7	-	**24.7**	-	**10.1**	23.5	**37.1**	**43.7**
		Ours ($\mathcal{L}_{tot}$)	78.3	**30.1**	78.0	1.7	**0.1**	24.1	**12.0**	14.6	79.7	-	79.1	**51.4**	**15.5**	**74.4**	-	23.7	-	9.1	22.7	**37.1**	**43.7**
	ResNet101	Source Only	39.5	18.1	75.5	10.5	0.1	26.3	9.0	11.7	78.6	-	81.6	57.7	21.0	59.9	-	30.1	-	15.7	28.2	35.2	40.5
		AdaptSegNet (feat) [282]	62.4	21.9	76.3	-	-	-	11.7	11.4	75.3	-	80.9	53.7	18.5	59.7	-	13.7	-	**20.6**	24.0	-	40.8
		MinEnt [151]	73.5	29.2	77.1	7.7	0.2	27.0	7.1	11.4	76.7	-	82.1	57.2	**21.3**	69.4	-	29.2	-	12.9	27.9	38.1	44.2
		SAPNet [366]	81.7	33.5	75.9	-	-	-	7.0	6.3	74.8	-	78.9	52.1	**21.3**	75.7	-	**30.6**	-	10.8	28.0	-	44.3
		MaxSquare IW [325]	78.5	34.7	76.3	6.5	0.1	**30.4**	12.4	12.2	**82.2**	-	**84.3**	**59.9**	17.9	80.6	-	24.1	-	15.2	31.2	40.4	46.9
		Ours ($\mathcal{L}'_{tot}$)	64.4	25.5	77.3	**14.3**	**0.9**	29.6	**21.2**	**24.2**	76.6	-	79.7	53.7	15.5	79.7	-	11.0	-	11.0	**35.2**	38.7	44.2
		Ours ($\mathcal{L}_{tot}$)	**88.3**	**42.2**	**79.1**	7.1	0.2	24.4	16.8	16.5	80.0	-	**84.3**	56.2	15.0	**83.5**	-	27.2	-	6.3	30.7	**41.1**	**48.2**

of 5×10^{-4}. Following [325], we also randomly apply mirroring and gaussian blurring for data augmentation during the training stage. To accommodate for GPU memory limitations, we resize images from the GTA5 dataset up to a resolution of 1280×720 px, as done by [282]. SYNTHIA images are instead kept to the original size of 1280×780 px. As for the target Cityscapes dataset, training unlabeled images are resized to 1024×512 px, whereas the results of the testing stage are reported at the original image resolution (2048×1024 px). We use a batch size of 1 and the hyper-parameters are tuned by resorting to a small subset of labeled target data that we set aside from the original target training set and reserve only for parameter selection. As evaluation metric, we employ the mean Intersection over Union (mIoU). The entire model is developed using PyTorch and trained with a single GPU. The code is available at `https://lttm.dei.unipd.it/paper_data/UDAclustering/`.

9.3.3 Results

We evaluate the performance of our approach to two widely used synthetic-to-real adaptation scenarios, namely the GTA5 → Cityscapes and SYNTHIA → Cityscapes benchmarks. Table 9.4 reports the numerical results of the experimental evaluation. We compare our model to several state-of-the-art methods, which, similarly to our approach, resort to a direct or indirect form of feature-level regularization and distribution alignment to achieve domain adaptation. With *source only* we indicate the naïve fine-tuning approach, in which no form of target adaptation assists the standard source supervision.

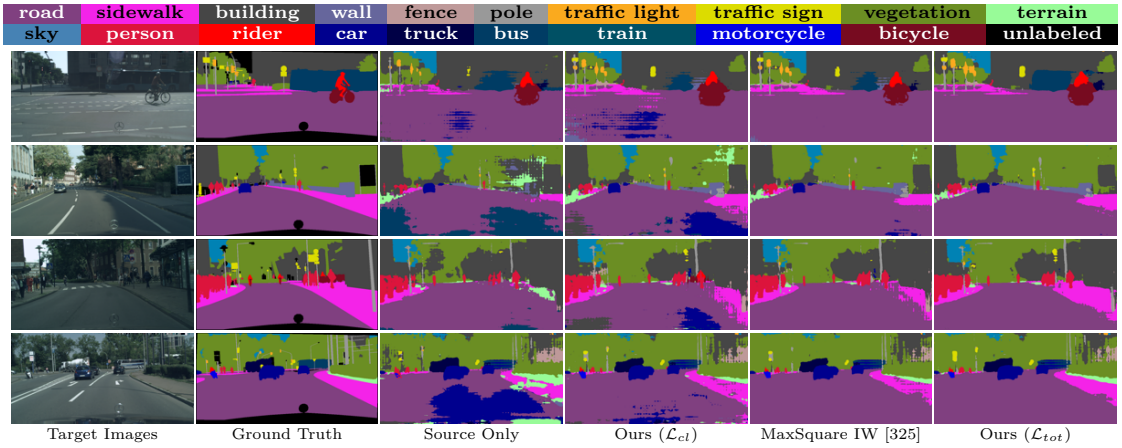


Figure 9.6: Semantic segmentation of some sample scenes from the Cityscapes validation dataset when adaptation is performed from the GTA5 source dataset and the DeepLab-V2 with ResNet-101 backbone is employed.

9.3.3.1 GTA5 → Cityscapes

For the GTA5 → Cityscapes and ResNet-101 configuration, our approach shows state-of-the-art performance in feature-level UDA for semantic segmentation, achieving 45.3% of mIoU, which is further boosted up to 45.9% by the entropy minimization objective. By looking at Table 9.4, we observe about 9% increase over the *source only* baseline, with the improvement well distributed over all the classes. A similar behavior can be noted when switching to the less performing VGG-16 backbone: we achieve 33.7% mean IoU with $\mathcal{L}'_{tot}$ (*i.e.*, without the entropy minimization objective) and 34.2% with $\mathcal{L}_{tot}$ (*i.e.*, with all components enabled) starting from the 25.4% of the baseline scenario without adaptation.
When compared with other approaches, our method performs better than standard feature-level adversarial techniques [273,275,282,366]. For example, with the VGG-16 backbone there is a gain of 4.5% with respect to [275]. This proves that a more effective class-conditional alignment has been ultimately achieved in the latent space by our approach. Due to the similar regularizing effect over feature distribution and comparable ease of implementation, we also compare our framework with some entropy minimization and self-training techniques [151, 323, 325], further showing the effectiveness of our adaptation strategy even if the gap here is a bit more limited. With both backbones, our novel feature level modules ($\mathcal{L}'_{tot}$) perform better than *MaxSquare IW* [325]. Moreover, adding the entropy minimization objective from [325] to $\mathcal{L}'_{tot}$ provides a slight but consistent improvement. It is worth noting that our method does not rely on additional trainable modules (*e.g.*, adversarial discriminators [273, 275, 282, 366]) and the whole adaptation process is end-to-end, not requiring multiple separate steps to be re-iterated (*e.g.*, pseudo-labeling in self-training [323]). Moreover, being focused solely on feature level adaptation, it could be easily integrated with other adaptation techniques working at different network levels, such as the input (*e.g.*, generative approaches) or output (*e.g.*, self-training), as shown by the addition of the output-level entropy-minimization loss.

Figure 9.6 displays some qualitative results on the Cityscapes validation set of the adaptation process when the ResNet-101 backbone is used. We observe that the introduction of the clustering module is beneficial to the target segmentation accuracy with respect to the *source only* case. Some small prediction inaccuracies remain, that are corrected with the introduction of the orthogonality, sparsity and entropy modules in the complete framework. By looking at

Table 9.5: Ablation results on the contribution of each adaptation module in the GTA5 to Cityscapes scenario and with ResNet-101 as backbone.

$\mathcal{L}_{cl}$	$\mathcal{L}_{or}$	$\mathcal{L}_{sp}$	$\mathcal{L}_{em}$	mIoU
				37.0
✓				42.3
	✓			43.2
		✓		43.7
			✓	44.8
✓	✓	✓		45.3
✓	✓	✓	✓	45.9

the last two columns, we also notice that our entire framework shows an improvement over the individual entropy-minimization like objective from [325], which is reflected in a better detection accuracy both on frequent (*e.g. road*, *vegetation*) and less frequent (*e.g. traffic sign*, *bus*) classes.

9.3.3.2 SYNTHIA → Cityscapes

To further prove the efficacy of our method, we evaluate it on the more challenging SYNTHIA → Cityscapes benchmark, where a larger domain gap exists. Once more, our approach proves to be successful in performing domain alignment with both ResNet-101 and VGG-16 backbones, reaching state-of-the-art results for feature-level UDA in both configurations (see Table 9.4). When ResNet-101 is used, the mIoU* on the 13 classes setting is pushed up to 48.2% from the original 40.5% of *source only*, while the VGG-16 scenario witnesses an even more improved performance gain of almost 11% over the no adaptation baseline till a final value of 43.7%. Differently from the GTA5 → Cityscapes case, here the contribution of the entropy-minimization module varies for the two backbones. The induced benefit is absent with VGG-16, since the clustering, orthogonality and sparsity jointly enforced already carry the whole adaptation effort. Besides, even the $\mathcal{L}_{em}$ objective alone (*i.e.*, *MaxSquares IW* $^{(r)}$ [325]) displays quite limited gain over the no adaptation baseline. On the contrary, the regularizing effect of the entropy objective is strongly valuable in case the ResNet-101 backbone is used. Yet, the combination of all modules together actually provides a noticeable boost over both the entropy and feature-level modules separately applied. As for the GTA5 scenario, our model shows better performance than feature-level adversarial adaptation [273,277,282,366] and output-level approaches [151,323] comparable in computational ease.

9.3.3.3 Ablation Study

To verify the robustness of the framework, we perform an extensive ablation study on the adaptation from GTA5 to Cityscapes with ResNet-101 as backbone. First, we examine the contribution of each loss to the final mIoU; then, we investigate the effect of each novel loss component.

The contribution of each loss to the adaptation module is shown in Table 9.5. Every loss component largely improves the final mIoU results from 37.0% of the *source only* scenario up to a maximum of 44.8%. Combining the 3 novel modules of this work, we achieve a mIoU of 45.3%, which is higher than all the losses alone, but lower than our complete framework with all the losses enabled (45.9%).

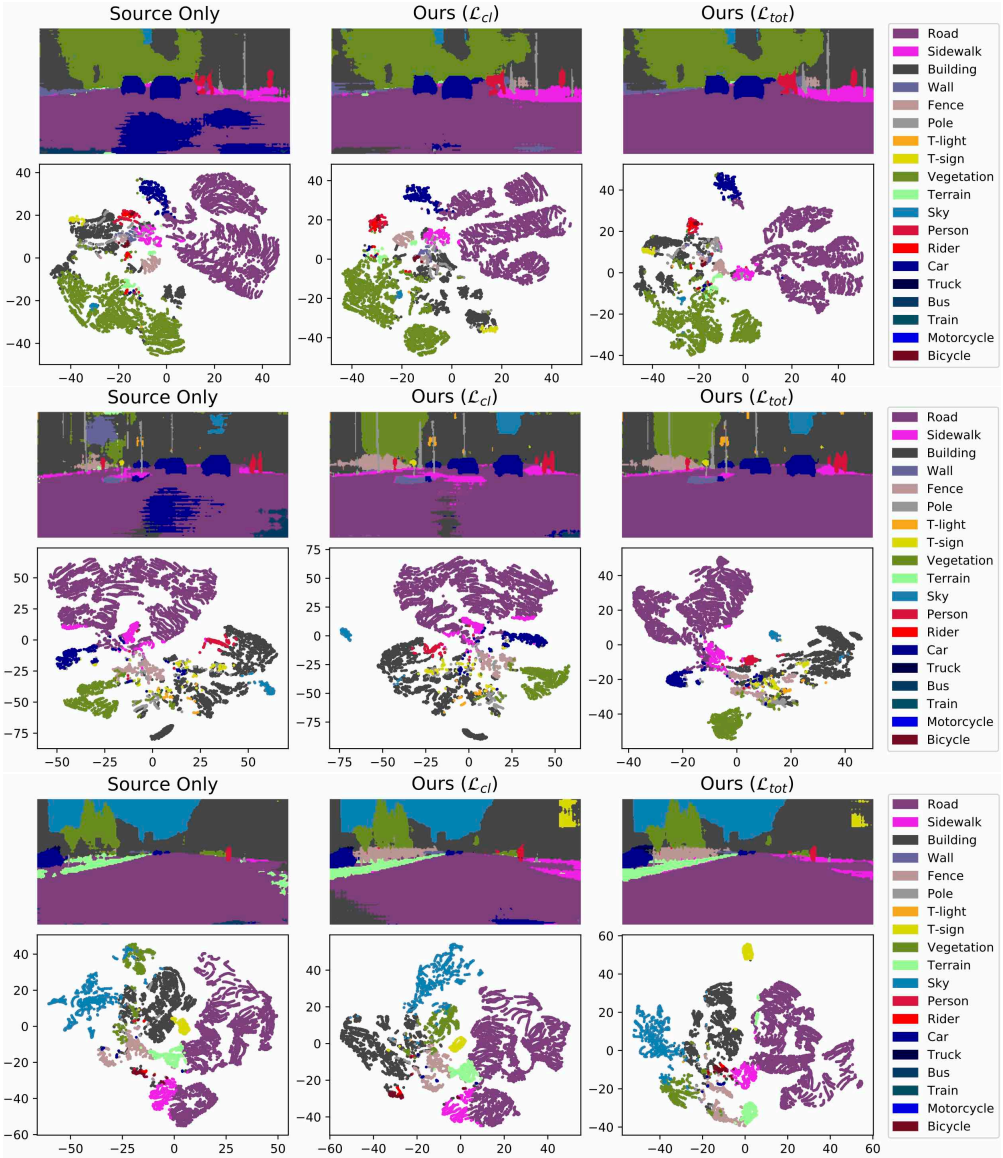


Figure 9.7: T-SNE computed over features of single images from the Cityscapes validation set when adapting from GTA5.

Clustering Loss. The effect introduced by the clustering objective ($\mathcal{L}_{cl}$) is investigated by means of the t-SNE tool [367] on features extracted from a sample image of the Cityscapes validation set. The results are reported in Figure 9.7. We extract all feature vectors from a single target image and reduce their dimensionality from 2048 to 2, so that we able to visualize their disposition. Each single feature instance is then associated to a semantic class from the ground-truth segmentation map, with the categorization expressed by the standard color map we jointly report. To allow for the analysis of the aggregating effect of the clustering module, in Figure 9.7 we show the t-SNE plots in the *source only* scenario, when only $\mathcal{L}_{cl}$ is enabled and when all adaptation modules are turned on ($\mathcal{L}_{tot}$). The effect brought by the clustering constraint is twofold. From one hand, we can see how features of the same class are more tightly clustered when $\mathcal{L}_{cl}$ is enabled, effect which is even further amplified by the class-conditional structural regularization provided by the other components of our work (see $\mathcal{L}_{tot}$). For instance, this is particularly visible in *road* and *vegetation* in row 2. From the other hand, we can appreciate that features belonging to different classes are more easily spaced apart, as shown in *car* or *person* in row 1 or in *traffic light* in row 3.

Orthogonality Loss. We investigate the contribution of the orthogonality constraint $\mathcal{L}_{or}$ via a similarity score defined as an average class-wise cosine similarity measure. The cosine distance is first computed for every pair of feature vectors from a single target image. Then, the average values are taken over all features from the same class to get a score for each pair of semantic classes. The final values are computed by averaging over all images from the Cityscapes validation set.

In Figure 9.8 we analyze the orthogonalizing action in three different configurations and we can clearly see that $\mathcal{L}_{or}$ causes the similarity score to significantly increase on almost all the classes (higher similarity reflects into lower orthogonality). To show the effect of the orthogonality constraint alone, we compare the *source only* similarity score and the scenario with only $\mathcal{L}_{or}$ enabled (Figure 9.8a). To investigate its effect when all the loss components are enabled, we compared the similarity scores of the full approach with respect to *source only* (Figure 9.8b) and to the case where all components but the $\mathcal{L}_{or}$ are enabled (Figure 9.8c). The results are robust and coherent in showing an increased similarity score in the configurations where $\mathcal{L}_{or}$ is active.

The same considerations are visible in Figure 9.9 in which the matrices of class-wise similarity scores are reported for the three aforementioned scenarios. In particular, we can see how the diagonal is much brighter (*i.e.*, high similarity for classes with themselves) when $\mathcal{L}_{or}$ is added, while off-diagonal entries are darker.

Sparsity Loss. To understand the efficacy of the sparsity loss ($\mathcal{L}_{sp}$) we compute the sparsity scores as the fraction of activations in normalized feature vectors close to 0 or 1. Closeness is quantified as being distant from 0 or 1 less than a threshold, which we set to 10^{-4}. In Figure 9.10 we plot the histogram distribution of all normalized feature activations with bin size set to 0.05 in linear scale and in log scale. Here, we can observe that adding $\mathcal{L}_{sp}$ leads to a greater number of occurrences of activations within 0 and 0.1 and within 0.95 and 1 than in the case without sparsity constraint. In the middle, instead, the opposite is true. We refer the reader to Eq. (9.16) to certify that this is indeed what the sparsity loss was aiming to achieve. Namely, the sparsity constraint reduces class-wise the number of active feature channels pushing them either toward 0 (inactive features) or towards 1 (active features). As for the similarity scores, the sparsity measures for a single target image are obtained through averaging over all feature vectors from the same class. The final results correspond to the mean values over the entire Cityscapes validation set. From Figure 9.10 we can appreciate that $\mathcal{L}_{sp}$ effectively achieves higher values of sparseness for all the classes.

Ultimately, for more immediate visualization, we plot in the third image of Figure 9.10 the difference of the sparsity distributions with and without $\mathcal{L}_{sp}$. We can more easily verify that

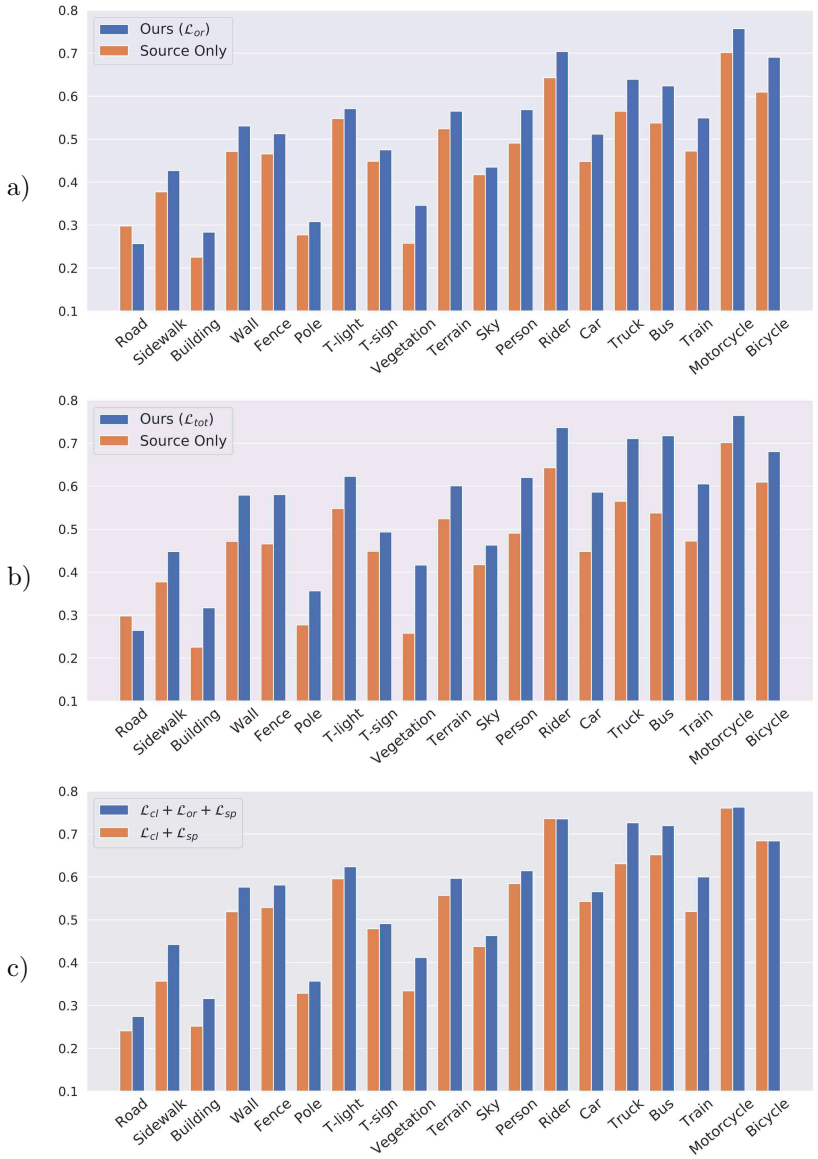


Figure 9.8: Similarity scores computed over all the images on the Cityscapes validation set when adapting from GTA5 to analyze the effect of the orthogonality constraint.

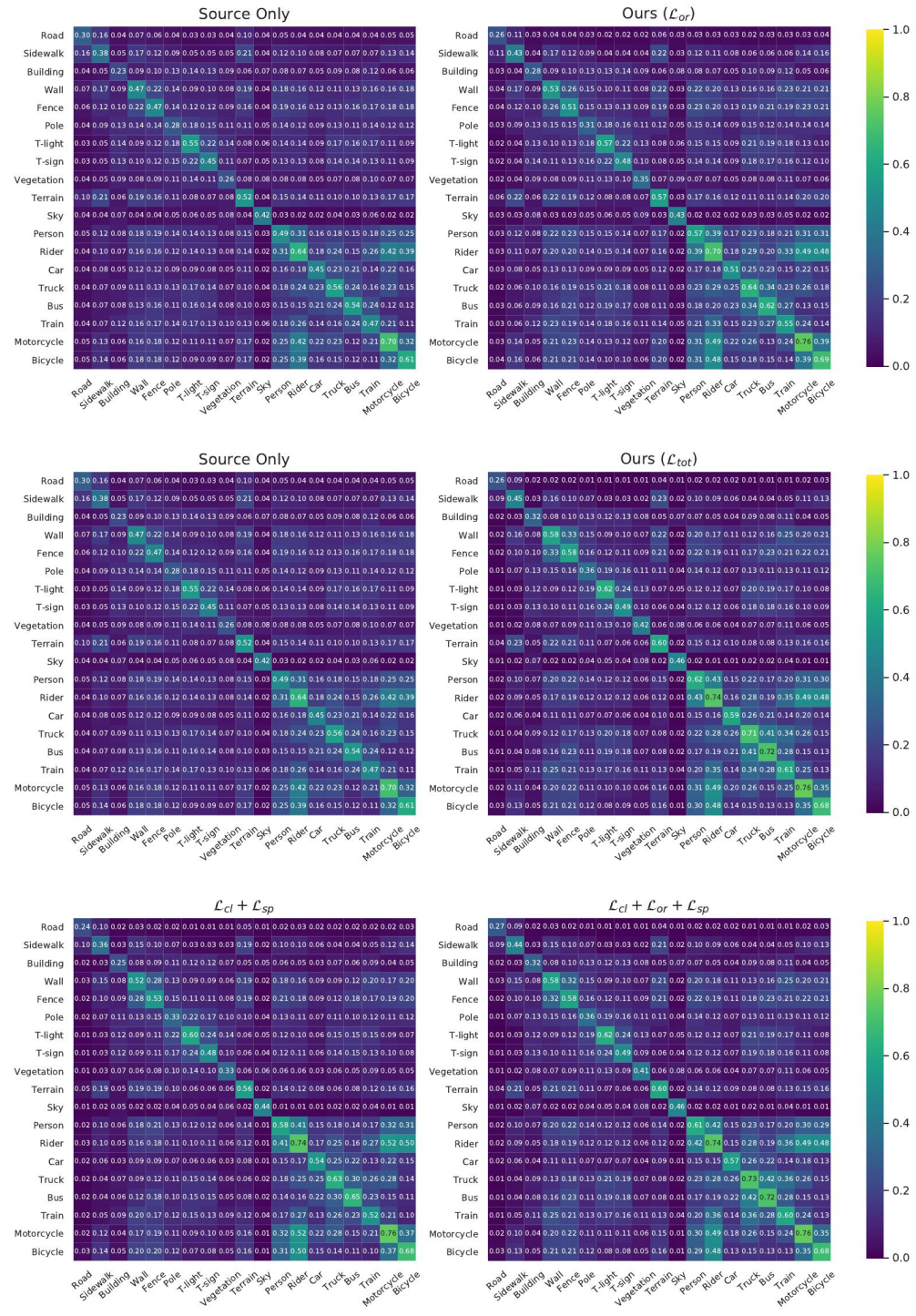


Figure 9.9: Class-wise similarity scores computed over images on the Cityscapes validation set when adapting from GTA5.

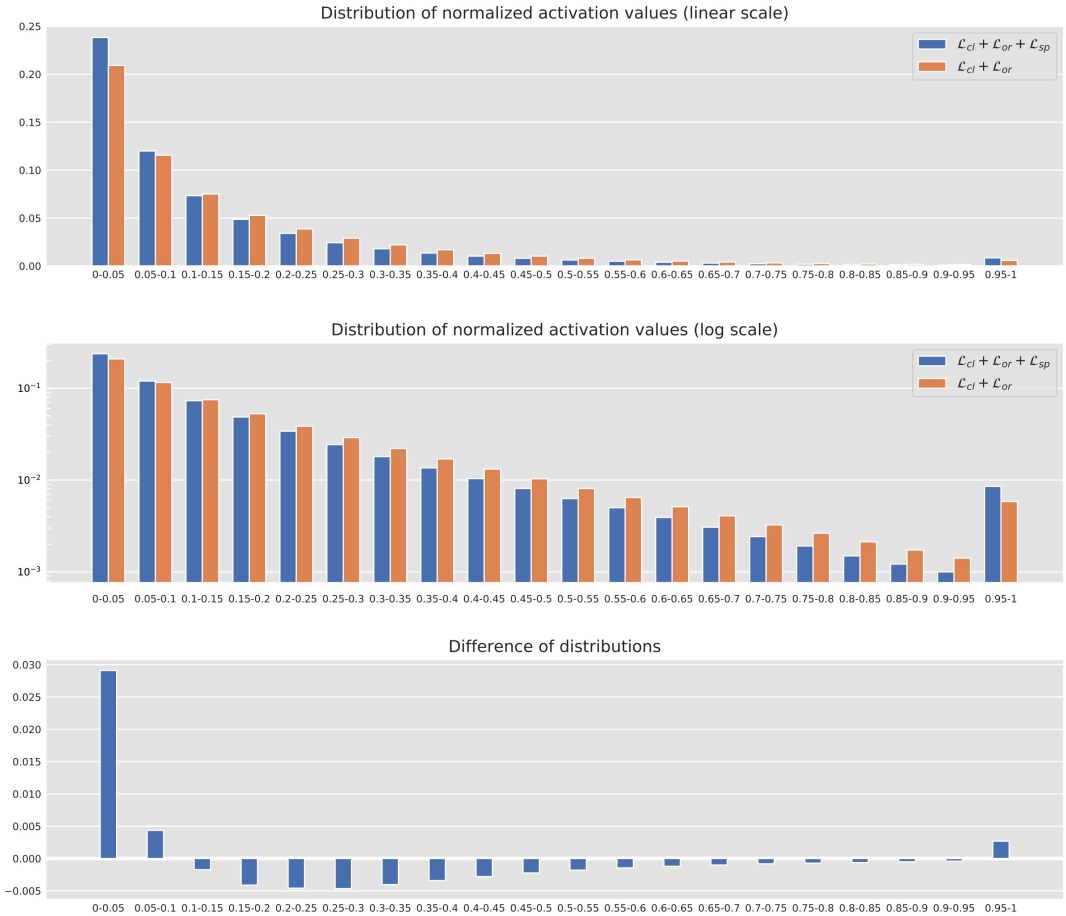


Figure 9.10: Analysis of the distribution of feature activations computed over all the images on the Cityscapes validation set when adapting from GTA5.

extremely low (in closest range to 0) and extremely high (in closest range to 1) bins have positive values while middle-range bins have negative values.

9.4 Feature-Level Regularization with Improved Prototypes Extraction

The last contribution of this chapter extends previous considerations on feature-level adaptation [31] and builds on top of our previous work [32]. By employing latent-space shaping objective, our aims are to promote class-aware features extraction and features invariance between source and target domains. Such improved regularity of the latent space has, in fact, shown to promote generalization properties, leading to statistical alignment between the source and target distributions when regularization is jointly applied over both domains [31, 32].

First of all, a clustering objective groups feature vectors of each class tightly around their prototypical representation. Second, a perpendicularity constraint over the class prototypes promotes disjoint filter activation sets across different semantic categories. Finally, a regularization-based norm alignment objective promotes uniform vector norms across source and target representations, while jointly inducing progressively increased norm values. This, together with the perpendicularity constraint, is able to reduce the entropy associated with the feature vector channel activations.

We remark that the proposed techniques require the generation of accurate class prototypes and the imposition of a strong relationship between predicted segmentation maps and feature representations. Hence, we additionally develop a novel strategy to propagate semantic information from the labels to the lower-resolution feature space (annotations downsampling).

We move from our previous work [32], which already achieved state-of-the-art results on feature-level UDA in semantic segmentation. Compared to it, we introduced several novel contributions. The computation of prototypes and feature vector extraction were refined. The first now considers the prototype trajectory evolution for a better estimation (Section 9.4.1-B), while the second exploits target information to reduce the domain shift (Section 9.4.1-C), additionally a class-weighting scheme is used in the source supervision (Section 9.4.1-A). Then, each of the three proposed space-shaping constraints was improved and additional ablation studies were performed both for the approach (Section 9.4.6) and for the evaluation metric (Section 9.4.4). In particular, the clustering objective was modified to be more resilient to outliers (Section 9.4.2.1); the perpendicularity constraint now accounts for classes not present in the current batch (Section 9.4.2.2); the norm augmentation now ignores low-activated channels (Section 9.4.2.3). Finally, extensive experiments were conducted on many different scenarios. The results are reported on 4 backbones and 6 setups (2 synthetic-to-real and 4 real-to-real). Additional results using the unlabeled Cityscapes coarse set [342] are reported, showing significant performance gains (see Table 9.6).

9.4.1 Problem Setup

First, we overview our setup, detailing the mathematical notation used throughout this section. We start by denoting the input image space as $\mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$ and the associated output label space as $\mathcal{Y} \subset \mathcal{C}^{H \times W}$, where H and W represent the spatial dimensions and $\mathcal{C}$ the set of classes. Furthermore, we assume to have a training set $\mathcal{T} = \mathcal{T}^s \bigcup \mathcal{T}^t$, where $\mathcal{T}^s = \{(\mathbf{X}_n^s, \mathbf{Y}_n^s)\}_{n=1}^{N_s}$ contains labeled samples $(\mathbf{X}_n^s, \mathbf{Y}_n^s) \in \mathcal{X}^s \times \mathcal{Y}^s$ originated from a supervised source domain, while a second set of unlabeled input samples $\mathcal{T}^t = \{\mathbf{X}_n^t\}_{n=1}^{N_t}$ are drawn from a target domain ($\mathbf{X}_n^t \in \mathcal{X}^t$). We transfer the knowledge of semantic segmentation learned on the source domain to the unsupervised target domain (*i.e.*, without any label on the target set). Superscripts s and t specify the domain: source and target, respectively.

As done by most recent approaches for semantic segmentation, we assume a task model $S = D \circ E$ based on an encoder-decoder architecture *i.e.*, made by the concatenation of two logical blocks: the encoder network E, consisting of the feature extractor, and a decoder network D, which is the actual classifier producing the segmentation map. We denote the features extracted from a generic input image $\mathbf{X}$ as $E(\mathbf{X}) = \mathbf{F} \in \mathbb{R}_{0+}^{H' \times W' \times K}$, where K refers to the number of channels and $H' \times W'$ to the low-dimensional latent space spatial resolution. Given the structure of encoder-decoder convolutional segmentation networks, we can assume that each class is mapped to a reference representation in the latent space, that should be as invariant as possible to the domain shift. The techniques that will be introduced in Section 9.4.2 enforces this goal by comparing the extracted features with some *prototypes* of the various classes. In the remainder of this section we show how to associate feature vectors to semantic classes and how to compute the prototypes.

A. Weighted Histogram-Aware Downsampling. Since the spatial information of an image is mostly preserved while its content travels through an encoder-decoder network, we can infer a strict relationship between any feature vector (*i.e.*, the vector of features associated to a single spatial location within the feature tensor) and the semantic labeling of the corresponding image region.

Therefore, the first step of the extraction process is to identify a way to propagate the labeling information to latent representations (decimation), preserving the semantic content of the image region (window) associated to each feature vector. Otherwise, the generation of erroneous associations would significantly impair the estimation objective. For this task, we design a non-linear pooling function: instead of computing a simple subsampling (*e.g.*, nearest neighbor), we compute a weighted frequency histogram over the labels of all the pixels in the window corresponding to a low-resolution feature location. The weights are inversely proportional to the class-frequency in the source training dataset. Such histograms are then used to select appropriate classification labels for the downsampled windows, producing source feature-level label maps $\{\mathbf{I}_n^s\}_{n=1}^{N_s}$. For what concerns the computation of the target counterparts ($\{\mathbf{I}_n^t\}_{n=1}^{N_t}$) see Section 9.4.1-C and we remark that each $\mathbf{I}_n^{s,t} \in \mathcal{C}^{H' \times W'}$. Specifically, the choice is made by selecting the label corresponding to the frequency peak in each window, only if such peak is distinctive enough, *i.e.*, if any other peak is smaller than T_h times the biggest one (in a similar fashion to the orientation assignment step in the SIFT feature extractor [368]). Empirically, we set $T_h = 0.5$. A key feature of this technique is its ability to introduce void-class samples when a considered window cannot be assigned to a unique class, *i.e.*, it contains mixed classification labels.

B. Prototype Extraction. Once computed, the feature-level label maps $\{\mathbf{I}_n^{s,t}\}_{n=1}^{N_{s,t}}$ can be used to extract the set $\mathcal{F}_c^{s,t}$ of feature vectors belonging to a generic class $c \in \mathcal{C}$ in a training batch $\mathcal{B}$:

$$\mathcal{F}_c^{s,t} = \left\{ \mathbf{F}_n^{s,t}[h,w] \in \mathbb{R}_{0+}^K \mid \mathbf{I}_n^{s,t}[h,w] = c, \forall n \in \mathcal{B} \right\}, \tag{9.17}$$

where $[h,w]$ denote all possible spatial locations over a feature map, *i.e.*, $0 \leq h < H'$ and $0 \leq w < W'$. Exploiting this definition, we can identify the set of all feature vectors in batch $\mathcal{B}$ as the union $\mathcal{F}^{s,t} = (\bigcup_c \mathcal{F}_c^{s,t}) \cup \mathcal{F}_v^{s,t}$ where $\mathcal{F}_v^{s,t}$ are the sets of void-class samples. The class-wise sets are then used to estimate the per-batch class prototypes on labeled source data by simply computing their centroids:

$$\mathbf{p}_c[i] = \frac{1}{|\mathcal{F}_c^s|} \sum_{\mathbf{f} \in \mathcal{F}_c^s} \mathbf{f}[i] \quad \forall i, \;\; 1 \leq i \leq K. \tag{9.18}$$

Finally, to reduce estimation noise and obtain more stable and reliable prototypes, we apply

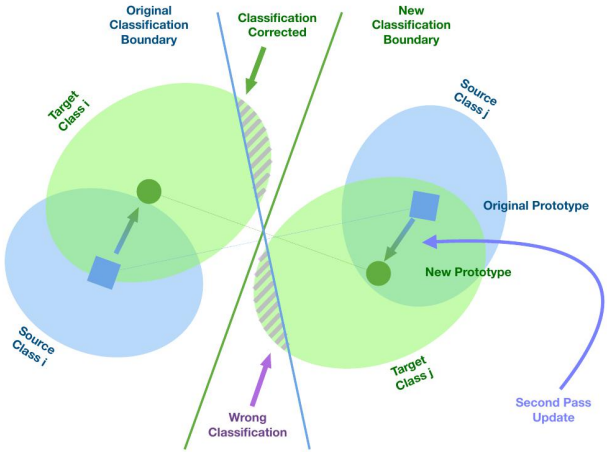


Figure 9.11: Visual representation of our two-pass feature vector classification strategy. The initial source-based classification (in blue) can lead to erroneously classified target samples (purple shaded areas). This problem is tackled by computing target prototypes as the centroids of the partitioned vectors (notice the shift compared to the original source prototype), these prototypes are used as new classification centers (green boundary), producing a correct segmentation.

exponential smoothing:

$$\hat{\mathbf{p}}_c = \eta \hat{\mathbf{p}}'_c + (1-\eta)\mathbf{p}_c. \tag{9.19}$$

Where $\hat{\mathbf{p}}_c$ and $\hat{\mathbf{p}}'_c$ are the estimates of class c prototype respectively at current and previous optimization steps. We initialized $\hat{\mathbf{p}}_c = \mathbf{0}$ and empirically set $\eta = 0.8$. This strategy allows us to keep track of classes that are not present in the current batch of source samples (in this case we set $\eta = 1$ to propagate the previous estimate), allowing for a more robust prototype estimation.

C. Feature pseudo-labeling. While the histogram strategy can be seamlessly extended to be used with pseudo-labels (*i.e.*, network estimates for the unlabeled target samples, as was our strategy in the previous work [32]), this approach can introduce instability in the training procedure. To avoid such issue, we devise a novel way of extracting the target feature-level label maps $\{\mathbf{I}^t_n\}_{n=1}^{N_t}$.

Our strategy exploits the euclidean distance in the latent space, computing a clustering of the feature vectors around their prototype (see Figure 9.11). More in detail, we compute an initial classification exploiting the prototypes computed over the source labeled data, which, due to the domain shift, will not be adequately representative of the target distribution:

$$\begin{aligned} \widetilde{\mathcal{F}}^t_c &= \{\mathbf{F}^t_n[h,w] \;\text{ if } \\ &\qquad \sigma_c(-||\mathbf{F}^t_n[h,w] - \hat{\mathbf{p}}_c||) > 0.5 \;\forall n \in \mathcal{B}\} \\ \mathbf{p}^t_c[i] &= \frac{1}{|\widetilde{\mathcal{F}}^t_c|} \sum_{\mathbf{f} \in \widetilde{\mathcal{F}}^t_c} \mathbf{f}[i] \qquad \forall i, \;\; 1 \leq i \leq K. \end{aligned} \tag{9.20}$$

Where $\sigma_c(\cdot)$ is the softmax function computed over the classes. Then, we refine the classification keeping only those vectors that have a high classification confidence according to a probability distribution attained through a softmax function:

$$\mathbf{I}^t_n[h,w] = \begin{cases} c & \sigma_c(-||\mathbf{F}^t_n[h,w] - \mathbf{p}^t_c||) > 0.5 \\ void & \text{otherwise} \end{cases} \tag{9.21}$$

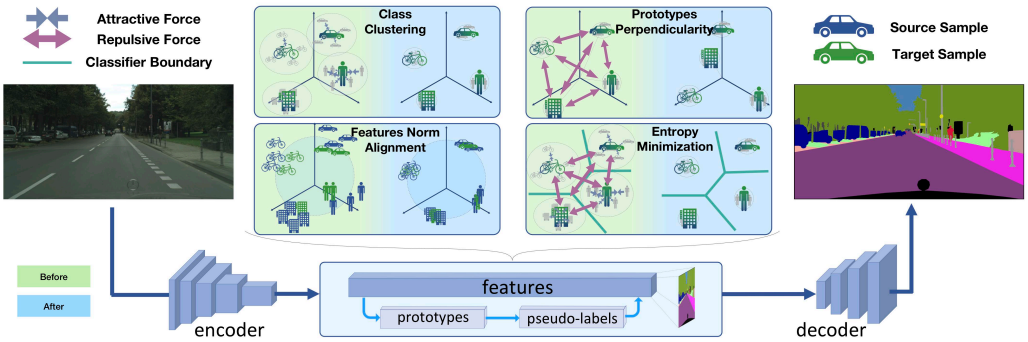


Figure 9.12: Visual summary of our strategy. Features are associated to semantic classes and prototypes are computed from them (9.4.1). The three proposed space shaping constraint are: Class Clustering, Prototypes Perpendicularity, Norm Alignment and Enhancement. Furthermore we apply also entropy minimization [325].

9.4.2 Proposed Latent-Level Constraints

In this section, we provide a detailed description of our approach, highlighting the key differences with respect to our previous work. Our investigation moves from the fact that the discriminative effect acquired by the model with the source supervised cross-entropy objective may not be propagated to the target domain due to the distribution shift. To tackle such problem, in [32] we proposed to use additional space-shaping objectives to increase the network generalization capability, therefore improving robustness to distribution shifts from the original source training data. In particular, we added three feature-space shaping constraints to the standard source-supervision ($\mathcal{L}^s_{CE}$), whose combined effect can be mathematically expressed by:

$$\mathcal{L} = \mathcal{L}^s_{CE} + \lambda_C \cdot \mathcal{L}^{s,t}_C + \lambda_P \cdot \mathcal{L}^s_P + \lambda_N \cdot \mathcal{L}^{s,t}_N. \tag{9.22}$$

Here, $\mathcal{L}_C$ represents the clustering objective acting on the feature vectors (Section 9.4.2.1), $\mathcal{L}_P$ the perpendicularity constraint applied to class prototypes (Section 9.4.2.2) and $\mathcal{L}_N$ the norm alignment goal (Section 9.4.2.3). For ease of notation, Eq. (9.22) reports each loss term once (the s, t superscript here indicates the sum of source and target loss instances). For an improved performance and to show that our approach can be applied on top of existing methods, we also extended our objective with the entropy minimization strategy proposed in [325], leading to $\mathcal{L}^+ = \mathcal{L} + \lambda_{EM} \cdot \mathcal{L}_{EM}$. By doing so, we also show that our space-shaping objectives provide a different and complementary effect on the feature vectors when compared to the entropy minimization constraint. An overview of the complete approach is reported in Figure 9.12.

9.4.2.1 Clustering of Latent Representations

Due to the domain shift between source and target domains, feature vectors originating from the two distributions are misaligned. This inevitably causes some incorrect classifications of target representations, in turn degrading the segmentation accuracy in the target domain. We introduce our first loss, a clustering objective over the latent space, to mitigate this problem, seeking for class-conditional alignment of feature distribution. We do so by exploiting the prototypical representations introduced in Section 9.4.1 and forcing source and target feature vectors to tightly assemble around them, regularizing the structure of the latent space and adapting representations into a common class-wise distribution.

Differently from the previous work, we define the clustering objective as the L1 distance

between feature vectors and their associated class-prototype. This results in a more stable training evolution and lower error rate in clustering, thanks to the outlier-rejecting properties of the L1 norm. In particular, due to the quadratic nature of MSE, outliers with distances greater than 1 have a strong push towards the clusters even when they should not. On the other hand, the L1 loss is stronger than MSE for close samples, which are more representative of each class, and is significantly gentler than L2 for distant outliers. The loss can be expressed mathematically as:

$$\mathcal{L}_C^{s,t} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{F}_c^{s,t}|} \sum_{\mathbf{f} \in \mathcal{F}_c^{s,t}} \frac{1}{K} \sum_{k=1}^{K} |\hat{\mathbf{p}}_c[k] - \mathbf{f}[k]|, \tag{9.23}$$

This loss has multiple purposes: first, to better cluster representations in the latent space in a supervised manner, thus reducing the probability of erroneous network classification. Second, to perform self-supervised clustering on target samples exploiting our two-pass pseudo-labeling strategy (see Section 9.4.1-C). Finally, to improve prototype estimates, since forcing tighter clusters will result in more stable batch-wise centroids, which will be closer to the moving-averaged prototypes.

9.4.2.2 Perpendicularity of Latent Representations

To further enhance the space-shaping action of the clustering objective, we introduce a prototype perpendicularity loss. The idea is to improve the segmentation accuracy by better separating the tight and domain-invariant clusters on both domains. By doing so, we allow the classifiers to increase the margin between decision boundaries and feature clusters, and, consequently, we reduce the likelihood of those boundaries to cross target high-density regions of the feature space (*i.e.*, regions populated by many target samples). We directly encourage a class-wise orthogonality property, not only increasing the distance among class clusters, but also encouraging channel-wise disjoint activations between different semantic categories.

To account for perpendicularity in the loss, we exploit the inner product in the euclidean space and its relationship with the angle θ between two vectors $\mathbf{j}$ and $\mathbf{k}$, *i.e.*, $\mathbf{j} \cdot \mathbf{k} = ||\mathbf{j}||\ ||\mathbf{k}|| \cos\theta$. Minimizing their normalized product is equivalent to maximizing the angle between them, since feature vectors have non-negative values. To capture this, we enforce cross-perpendicularity between any couple of prototypes:

$$\mathcal{L}_P^s = \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{c_i, c_j \in \mathcal{C}, i \neq j} \frac{\hat{\mathbf{p}}_{c_i}}{||\hat{\mathbf{p}}_{c_i}||} \cdot \frac{\hat{\mathbf{p}}_{c_j}}{||\hat{\mathbf{p}}_{c_j}||}. \tag{9.24}$$

Eq. (9.24) computes the sum of the cosines over the set of all couples of non-void classes. Thanks to the tight geometric relation between prototype estimates and feature vectors enforced by $\mathcal{L}_C^{s,t}$, the effect induced by the orthogonality constraint on the prototypes is propagated to the vectors associated to them. The net result is the application of the shaping action to all feature vectors of each class, thus promoting perpendicularity between all individual components of distinct clusters. The loss seeks to increase the angular distance between latent representations of separate classes, which is achieved when distinct sets of active feature channels are associated to distinct semantic categories.

In contrast to our previous paper [32], we compute the loss on the exponentially smoothed version of the prototypes (*i.e.*, from Eq. (9.19)). This guarantees that the space will be more evenly occupied by the classes, since all directions are considered in the computation of the loss, instead of considering only the ones in the current batch.

9.4.2.3 Latent Norm Alignment Constraint

The last constraint we propose acts on the norm of source and target feature vectors. In particular, we promote the extraction of latent representations with uniform norm values across domains. Our objective is twofold. First, we aim at increasing the classification confidence during target prediction, similarly to what achieved by adaptation strategies based on entropy minimization over the output space [151]. Second, we assist the perpendicularity loss by reducing the number of domain-specific feature channels exploited to perform classification. We argue, in fact, that by forcing the network to produce consistent feature norms, we reduce the number of channel activations switched on for only one of the two domains, as they would cause norm discrepancies. Moreover, to reduce the possible decrease in norm value during the alignment process, we introduce a regularization term that promotes norm increase. Differently from [32], here the norm objective is encoded as a relative difference with a regularization term inversely proportional to the norm value. This allows to obtain a value-independent loss where norm values higher than the target are less discouraged. Moreover, we introduce a *norm filtering* strategy to reduce the negative effects a careless increase in norm could imply. In particular, we suppress low channel activations, stopping the gradient flow through them and preventing the norm alignment procedure to increase their value, in contrast to what source supervision indicates. Formally, we define the loss term as:

$$\mathcal{L}_N^{s,t} = \frac{1}{|\mathcal{F}_*^{s,t}|} \sum_{\mathbf{f} \in \mathcal{F}_*^{s,t}} \frac{\left| (\bar{f}_s + \Delta_f) - ||\mathbf{f}|| \right|}{\bar{f}_s}, \tag{9.25}$$

where $\bar{f}_s$ is the mean of the feature vector norms computed from source samples in the previous optimization step, Δ_f dictates the regularization strength (experimentally tuned to 0.1) and $\mathcal{F}_*^{s,t}$ is a thresholded version of $\mathcal{F}^{s,t}$ where we set to 0 the low-activated channels of each feature vector, stopping the gradient propagation:

$$\begin{aligned} \mathcal{F}_*^{s,t} &= \{\phi(\mathbf{f}) \quad \forall \mathbf{f} \in \mathcal{F}^{s,t}\}, \\ \phi(\mathbf{f})_i &= \begin{cases} \mathbf{f}_i & \mathbf{f}_i \geq \frac{1}{K} \sum_{j=1}^{K} \mathbf{f}_i, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{9.26}$$

Feature vectors are pushed towards the same global average norm value, regardless of their labeling. This removes any bias generated by heterogeneous pixel-class distribution in semantic labels, which, for example, would cause the most frequent classes to show larger norm than the average. The constraint of Eq. (9.25) forces the inter-class alignment step, *i.e.*, it ensures that norms are progressively aligned throughout the training process towards a common value for all the classes, while guaranteeing the value does not decrease significantly. In other words, the goal value is computed on the source samples and it is the same for both datasets. An additional benefit of rescaling the loss by the norm target is that the loss gradients will be limited in magnitude and, therefore, more stable.

9.4.3 Implementation Details

Baseline Model. We used the common [31, 151, 282, 325, 369] DeepLabV2 network [4–6], with ResNet101 [152] as the backbone (with $K = 2048$ channels at the last level of the encoder) and stride 8. We pre-train the model following the same procedure as our previous work [32], and employing the same data augmentation techniques used during adaptation.

Training Procedure. We optimize the network using SGD with momentum of rate 0.9 and weight decay regularization of 5×10^{-4}. The learning rate follows a polynomial decay of power 0.9 starting from 2.5×10^{-4} over $250k$ steps, following [325]. A subset of the original training set was exploited as validation set for the hyper-parameters search in our loss terms. To reduce overfitting we employ various dataset augmentation strategies: random left-right flip; white point re-balancing $\propto \mathcal{U}([-75, 75])$; color jittering $\propto \mathcal{U}([-25, 25])$ (both applied independently over color channels) and random Gaussian blur $[323, 325]$. We used one NVIDIA Titan RTX GPU, with batch size of 2 (1 source and 1 target samples), training the network for $24,750$ steps (*i.e.*, 10 epochs of the Cityscapes-fine train split) and employing early stopping based on the validation set.

The code developed for this work is publicly available at the following link: `https://github.com/LTTM/LSR`.

9.4.4 Mean Adapted-to-Supervised Ratio Metric

In this section we introduce a novel measure, called mASR (*mean Adapted-to-Supervised Ratio*), in order to better evaluate the domain adaptation task than allowed by the usual mIoU.

The idea behind the new metric sparks from realizing that the mIoU is missing a key component to evaluate an adaptation method: *i.e.*, it does not account for the starting accuracy for the different classes in supervised training. In particular, the objective of domain adaptation is to transfer the knowledge learned on a source dataset to a target one, trying to get as close as possible to the results attainable through supervised learning on the target domain. We design mASR to capture the relative performance between an adapted architecture and its target supervised counterpart, which we identify as a reasonable upper bound. Therefore mASR focuses less on the absolute-term performance and more on the relative accuracy obtained by an adapted architecture when compared to traditional supervised training.

We compare the per-class IoU score of the adapted network for each $c \in \mathcal{C}$ (IoU^c_{adapt}) with the results of supervised training on target data (IoU^c_{sup}) and we compute mASR by:

$$\text{mASR} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{ASR}^c, \quad \text{ASR}^c \stackrel{def}{=} \frac{\text{IoU}^c_{adapt}}{\text{IoU}^c_{sup}} \cdot 100. \tag{9.27}$$

In mASR, the contribution of each class is inversely proportional to the capacity of the segmentation model to learn it in the supervised reference scenario, thus emphasizing the most challenging semantic categories and producing a more class-agnostic adaptation score. In this metric, higher means better and when the adapted network has the same performance as supervised training the score is 100%.

As an example, the mASR scores reported in the last two columns of Table 9.6 allow to identify at a glance the algorithms that more faithfully match the target performance.

To validate the new metric, we used as reference the supervised training on the Cityscapes dataset and compared it with the training on corrupted versions of the same dataset using the introduced mASR metric to evaluate the relative performance and so, indirectly, the domain shift introduced by the perturbations. In Figure 9.13 we identified 5 types of perturbations which are likely to be encountered by an agent moving outdoor (*i.e.*, Gaussian noise, motion blur, snow, fog, brightness) and we set 5 levels of noise intensity as defined by [370]. As expected, the higher is the noise intensity and the lower is the adaptation score computed by mASR. Furthermore, we can also have a hint of the most detrimental types of noise for adapting source knowledge: namely, Gaussian noise, snow, motion blur. This can help us identify which set of samples we

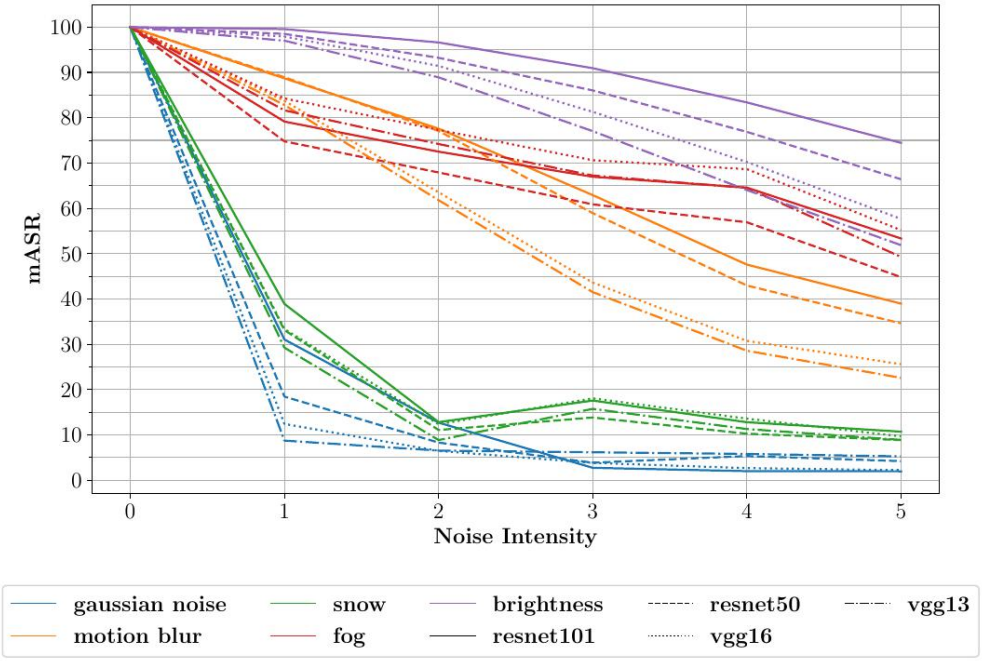


Figure 9.13: mASR score as a function of the injected noise intensity.

should consider more in order to obtain a reliable model capable of handling these situations. On the other hand, brightness and fog influence less the final results.

9.4.5 Results

In this section, we report the quantitative and qualitative results achieved by the proposed approach (LSR$^+$) and we compare it with several feature-level approaches ([31, 282, 366]), with some entropy minimization strategies ([151, 325]) that have a similar effect on feature distribution, and finally with the conference version of our work [32]. An ablation study and a discussion of the effects brought by the proposed loss terms is also presented.

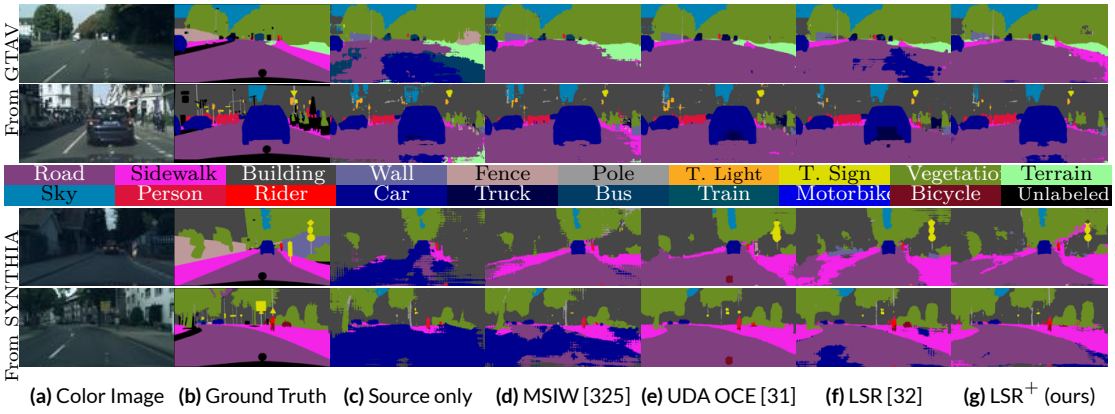
Since our method is trained end-to-end, it allows to seamlessly add other adaptation techniques, *e.g.*, entropy minimization or adversarial input or output level approaches. To prove such compatibility, we introduce an additional entropy-minimization loss [325] in our framework. We start from considering two widely used synthetic-to-real benchmarks and a standard ResNet-101 as backbone architecture obtaining the results shown in Table 9.6. Then, a real-to-real benchmark [277] has also been used (see Table 9.7). To further verify the robustness of our setup, in Table 9.8 we report some results using different backbones (*i.e.*, ResNet50, VGG16 and VGG13).

9.4.5.1 Adaptation from Synthetic Data to Cityscapes

When adapting source knowledge from the *GTA5* dataset to the Cityscapes one, our approach (LSR$^+$) achieves a mIoU of 46.9%, with a gain of 10% compared to the baseline and of 1% compared to the conference version (LSR) [32], thanks to the enhanced latent space regularization. In addition, it outperforms all competitors, with only the very recent works of [31] and [325] able

Table 9.6: Comparison of adaptation strategies in terms of IoU, mIoU and mASR (Section 9.4.5). Best in **bold**, runner-up underlined. mIoU[1] and mASR[1] restricted to 13 classes, ignoring the classes with same superscript.

Backbone	Setup	Configuration	Road	Sidewalk	Building	Wall[1]	Fence[1]	Pole[1]	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU	mIoU[1]	mASR	mASR[1]
ResNet101		Target only	96.5	73.8	88.4	42.2	43.7	40.7	46.1	58.6	88.5	54.9	91.9	68.7	46.2	90.7	68.8	69.9	48.8	47.6	64.5	64.8	-	100	100
	From GTAV	Source only [31]	71.4	15.3	74.0	21.1	14.4	22.8	33.9	18.6	80.7	20.9	68.5	56.6	27.1	67.4	32.8	5.6	7.7	28.4	33.8	36.9	-	54.0	-
		ASN (feat) [282]	83.7	27.6	75.5	20.3	19.9	27.4	28.3	27.4	79.0	28.4	70.1	55.1	20.2	72.9	22.5	35.7	8.3	20.6	23.0	39.0	-	56.9	-
		MinEnt [151]	84.4	18.7	80.6	23.8	23.2	28.4	36.9	23.4	83.2	25.2	79.4	59.0	**29.9**	78.5	33.7	29.6	1.7	29.9	33.6	42.3	-	61.9	-
		SAPNet [366]	88.4	**38.7**	79.5	29.4	**24.7**	27.3	32.6	20.4	82.2	32.9	73.3	55.5	26.9	82.4	31.8	41.8	2.4	26.5	24.1	43.2	-	63.1	-
		MaxSquareIW [325]	87.7	25.2	**82.9**	**30.9**	24.0	29.0	35.4	24.2	84.2	38.2	79.2	59.0	27.7	79.5	34.6	44.2	7.5	31.1	40.3	45.5	-	62.2	-
		UDA OCE [31]	**89.4**	30.7	82.1	23.0	22.0	29.2	37.6	**31.7**	83.9	37.9	78.3	60.7	27.4	**84.6**	37.6	44.7	7.3	26.0	38.9	45.9	-	67.3	-
		LSR [32]	87.7	32.6	82.6	29.1	23.0	28.5	36.1	28.5	**84.8**	41.8	**80.1**	59.4	23.8	76.5	**38.4**	**45.8**	7.1	28.5	40.1	46.0	-	67.7	-
		LSR$^+$ (ours)	88.9	26.6	82.0	21.0	24.4	**30.1**	**41.1**	27.0	84.7	**42.7**	**80.1**	**63.0**	26.4	83.1	30.4	44.3	**16.8**	**35.8**	**42.4**	**46.9**	-	**69.5**	-
		LSR$^+$ on *CS-full*	89.3	28.7	82.1	25.2	27.5	31.9	40.3	33.2	84.7	38.7	81.2	63.2	27.2	85.2	34.7	43.9	9.8	37.2	47.7	48.0	-	71.3	-
	From SYNTHIA	Source only [31]	17.7	15.0	74.3	10.1	0.1	25.5	6.3	10.2	75.5	-	77.9	57.1	19.2	31.2	-	31.2	-	10.0	20.1	30.1	34.3	41.7	44.6
		ASN (feat) [282]	62.4	21.9	76.3	-	-	-	11.7	11.4	75.3	-	80.9	53.7	18.5	59.7	-	13.7	-	**20.6**	24.0	-	40.8	-	52.5
		MinEnt [151]	73.5	29.2	77.1	7.7	0.2	27.0	7.1	11.4	76.7	-	82.1	57.2	21.3	69.4	-	29.2	-	12.9	27.9	38.1	44.2	51.1	56.3
		SAPNet [366]	81.7	33.5	75.9	-	-	-	7.0	6.3	74.8	-	78.9	52.1	21.3	75.7	-	30.6	-	10.8	28.0	-	44.3	-	56.0
		MaxSquareIW [325]	78.9	33.5	75.3	15.0	**0.3**	27.5	13.1	**16.7**	73.8	-	77.7	50.4	19.9	66.7	-	**36.1**	-	13.7	32.1	39.4	45.2	53.8	58.3
		UDA OCE [31]	**88.3**	**42.2**	79.1	7.1	0.2	24.4	**16.8**	16.5	80.0	-	**84.3**	56.2	15.0	**83.5**	-	27.2	-	6.3	30.7	41.1	48.2	54.3	60.9
		LSR [32]	81.0	36.9	**79.5**	13.4	0.2	28.7	9.0	16.1	79.1	-	81.7	57.9	**21.6**	77.2	-	35.3	-	14.2	35.4	41.7	48.1	56.5	61.6
		LSR$^+$ (ours)	82.6	38.4	**80.6**	**15.5**	**0.3**	**31.8**	6.7	16.3	**81.7**	-	82.5	**58.4**	20.2	81.3	-	32.7	-	15.3	**36.7**	**42.6**	**48.7**	**57.7**	**62.1**
		LSR$^+$ on *CS-full*	89.4	47.9	79.4	13.9	0.4	29.5	10.0	16.5	79.5	-	83.3	57.7	17.0	84.3	-	37.7	-	21.5	28.6	43.5	50.2	58.8	64.2



(a) Color Image **(b)** Ground Truth **(c)** Source only **(d)** MSIW [325] **(e)** UDA OCE [31] **(f)** LSR [32] **(g)** LSR$^+$ (ours)

Figure 9.14: Qualitative results on sample scenes taken from the *Cityscapes* validation split.

to get close to our result, while there is a quite relevant gap compared to all the other methods. Such improvement is quite stable across most per-class IoU scores, and is particularly evident in challenging classes, such as *terrain* and *t. light* where our strategy shows very high percentage gains, and on *train* where we significantly outperform the competitors by doubling the score of the second-best strategy.

Some qualitative results are reported in the top half of Figure 9.14. From visual inspection, we can verify the increased precision of edges in the *t. sign*, *t. light*, *pole* and *person* classes in both images. Furthermore, our approach is the only one to correctly classify the *bus* in the right of the first image as such (confused as *truck* by the other strategies). Importantly, we can also see the effects of our two-pass labeling (see Section 9.4.1-C) on the left of the top image (where part of the *fence* is correctly classified by our strategy, while being missed by all competitors) and of the second image (where LSR$^+$ significantly reduces the confusion between *sky* and the white building).

In the SYNTHIA to Cityscapes setup, LSR$^+$ surpasses its conference version (LSR) by about 1% of mIoU in the 16-classes setup and by 0.6% in the 13-classes one, achieving a final score of 42.6% and 48.7%, respectively. It also outperforms all the other competitors, with a slight

Table 9.7: Quantitative results on the Cross-City real-to-real benchmark. (r) indicates that the strategy was re-trained, starting from the official code. Best in **bold**, runner-up underlined.

Target City	Configuration	Road	Sidewalk	Building	T. Light	T. Sign	Vegetation	Sky	Person	Rider	Car	Bus	Motorbike	Bicycle	mIoU
Rome	Source only [325]	85.0	34.7	86.4	17.5	39.0	84.9	85.4	43.8	15.5	81.8	46.3	38.4	4.8	51.0
	Cross-City [277]	79.5	29.3	84.5	0.0	22.2	80.6	82.8	29.5	13.0	71.7	37.5	25.9	1.0	42.9
	ASN (feat) [282]	83.9	34.2	**88.3**	18.8	40.2	**86.2**	93.1	47.8	21.7	80.9	47.8	48.3	8.6	53.8
	MaxSquareIW [325] (r)	**86.2**	**37.8**	86.4	22.3	39.5	85.4	84.0	49.5	21.2	82.7	**55.3**	48.5	**9.5**	54.5
	UDA OCE [31] (r)	85.6	35.0	87.9	23.1	42.0	85.9	89.2	49.3	24.3	82.8	48.8	48.5	9.0	54.7
	LSR$^+$ (ours)	83.4	34.5	88.1	**29.0**	**44.5**	85.5	**93.9**	**51.9**	**31.3**	**83.2**	44.7	**51.5**	8.8	**56.2**
Rio	Source only [325]	74.2	42.2	84.0	12.1	20.4	78.3	**87.9**	50.1	25.6	**76.6**	40.0	27.6	17.0	48.9
	Cross-City [277]	74.2	43.9	79.0	2.4	7.5	77.8	69.5	39.3	10.3	67.9	**41.2**	27.9	10.9	42.5
	ASN (feat) [282]	76.2	44.7	84.6	9.3	**25.5**	**81.8**	87.3	55.3	32.7	74.3	28.9	**43.0**	27.6	51.6
	MaxSquareIW [325] (r)	**79.5**	50.7	84.5	**14.9**	17.7	80.8	85.7	54.5	29.6	75.1	37.0	40.6	24.5	51.9
	UDA OCE [31] (r)	78.9	48.5	**85.3**	14.2	24.4	81.3	87.0	55.9	36.2	74.3	29.7	41.8	27.9	**52.7**
	LSR$^+$ (ours)	**79.5**	**52.2**	83.7	10.2	23.1	79.3	82.3	**59.8**	**40.0**	75.0	23.0	**43.0**	**29.0**	52.3
Tokyo	Source only [325]	81.4	28.4	78.1	14.5	19.6	81.4	86.5	51.9	22.0	70.4	18.2	22.3	46.4	47.8
	Cross-City [277]	83.4	**35.4**	72.8	12.3	12.7	77.4	64.3	42.7	21.5	64.1	**20.8**	8.9	40.3	42.8
	ASN (feat) [282]	81.5	26.0	77.8	**17.8**	**26.8**	82.7	**90.9**	55.8	**38.0**	72.1	4.2	**24.5**	50.8	49.9
	MaxSquareIW [325] (r)	84.1	32.9	76.7	11.3	23.8	82.3	87.4	55.3	30.0	72.0	8.6	18.9	47.1	48.5
	UDA OCE [31] (r)	**85.0**	33.3	77.9	8.5	25.5	82.5	89.4	**56.1**	29.2	**72.4**	2.1	12.3	41.9	47.4
	LSR$^+$ (ours)	84.2	34.6	**78.2**	16.8	22.6	**83.3**	89.3	55.0	33.2	72.0	8.6	20.5	**52.2**	**50.0**
Taipei	Source only [325]	**82.6**	**33.0**	86.3	16.0	**16.5**	78.3	83.3	26.5	8.4	70.7	36.1	47.9	15.7	46.3
	Cross-City [277]	78.6	28.6	80.0	13.1	7.6	68.2	82.1	16.8	9.4	60.4	34.0	26.5	9.9	39.6
	ASN (feat) [282]	81.7	29.5	85.2	**26.4**	15.6	76.7	91.7	31.0	**12.5**	71.5	41.1	47.3	27.7	49.1
	MaxSquareIW [325] (r)	80.9	31.3	83.3	12.9	13.4	75.4	89.5	31.8	3.9	69.0	**44.3**	49.4	**33.3**	47.6
	UDA OCE [31] (r)	81.4	30.1	84.3	16.7	13.4	75.4	**91.9**	32.5	4.6	71.0	41.4	48.0	**33.3**	48.0
	LSR$^+$ (ours)	81.8	32.9	**86.8**	19.1	14.2	**79.3**	91.8	**35.1**	11.6	**72.8**	33.8	**58.7**	31.6	**50.0**

margin of 1% on average with respect to [31] and a larger one (more than 3%) with respect to all the other approaches. Finally, we also considered additional unlabeled samples from the Cityscapes dataset (*i.e.*, taking the coarsely-labeled split) and we refer to this dataset as *CS-full*). From Table 9.6), we can see that additional unlabeled data can further leverage the adaptation process.

Qualitative results are reported in the bottom half of Figure 9.14, where the overall increase in segmentation accuracy for many classes such as *car*, *road* and *sidewalk* is evident. In the first image (third row of Figure 9.14) we can see how LSR$^+$ is the only strategy to correctly classify both *rider* and *bike*, whereas other strategies even miss the *t. sign* in the foreground. Similarly, in the second image we note improvements on the prediction on such classes and, fundamentally, of the *road* in foreground (confused for *car* and *bycicle* by the competitors).

9.4.5.2 Adaptation from Cityscapes to Cross-City

In this subsection we discuss the contents of Table 9.7, where the performance on the Cross-City real-to-real benchmark is reported. This benchmark is comprised of 4 cities: Rome, Rio, Tokyo and Taipei. When evaluated on those setups, our strategy reaches an mIoU score of 56.2%, 52.3%, 50.0% and 50.0% surpassing the source only model by 5.2%, 3.4%, 2.2% and 3.7%, respectively. Importantly, our approach achieves consistent results across the setups (LSR$^+$ is the top scorer in 3 out of 4 setups and second in the remaining one) surpassing the average best competitor score by 0.5% mIoU (52.1% versus 51.6%). We remark that the best competitor changes depending on the setup, being [31], [31], [282] and [282] for Rome, Rio, Tokyo and Taipei,

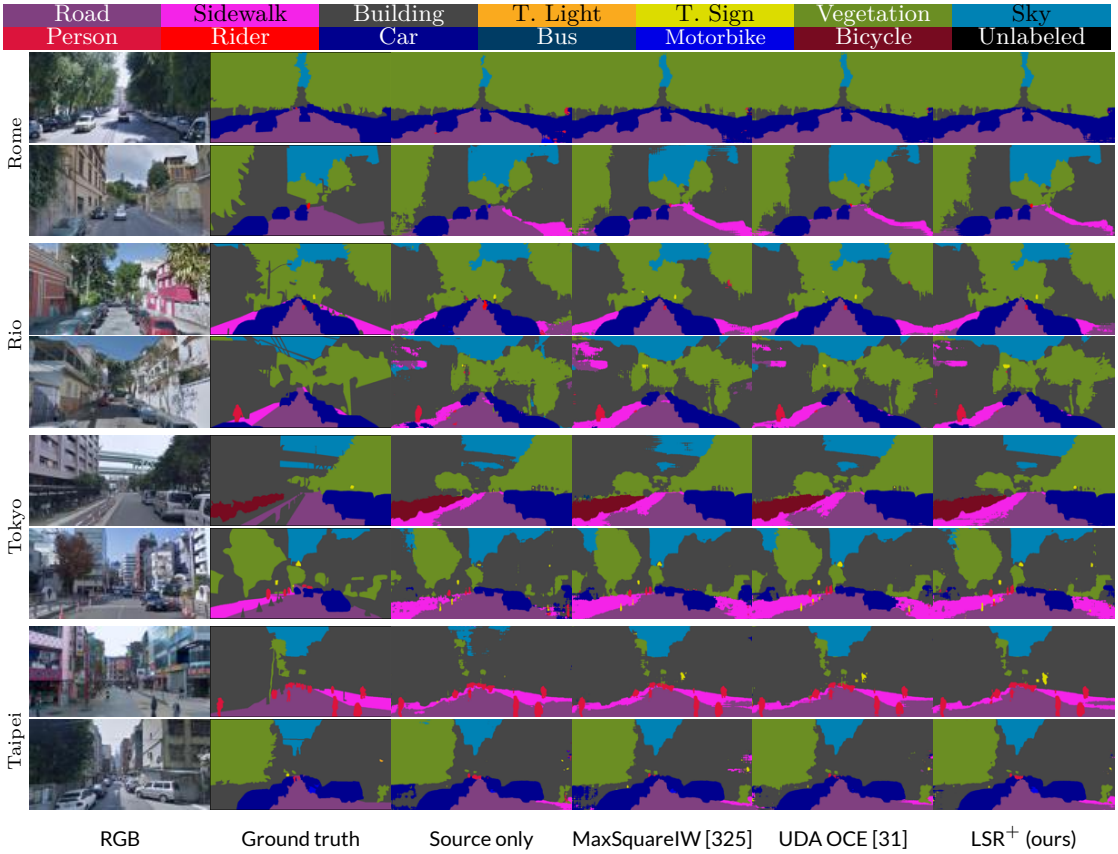


Figure 9.15: Qualitative results on the Cross-City benchmark.

respectively, underlining the unstable performance of many approaches usually associated with this benchmark.
Looking at the per-class IoU scores, we can see how our strategy significantly outperforms the competitors in *t. light* and *rider* in the Cityscapes→Rome setup (increase of 6% of IoU), in *person* and *rider* in the Cityscapes→Rio setup (increase of 4% of IoU) and in *motorbike* in the Cityscapes→Taipei setup (increase of 9.3% of IoU).

In Figure 9.15 we report some qualitative results on the Cross-City benchmark. Here we present two images for each city (Rome, Rio, Tokyo, Taipei) and compare our strategy with three other strategies (Source only, MaxSquareIW [325] and UDA OCE [31]).
From a visual inspection of the images we can see an overall increase in the discrimination of the object borders, particularly for classes such as *car*, *road*, *building*, *vegetation* and *person*.

In Rome we see how LSR$^+$ is the only strategy that correctly identifies the *rider* behind the cars in the second image. In Rio, our architecture significantly reduces the amount of confusion regarding the *building* on the left of the second image. Again, in Tokyo, we note how LSR$^+$ is the only technique able to recognize the *traffic sign* on the right of the first image. Finally, in Taipei, we see how our approach is the only to correctly identify the *person* and *motorcycle* in the second image.

Table 9.8: Additional quantitative results with multiple backbones, GTAV→Cityscapes setup. (r) indicates that the strategy was re-trained, starting from the official code.

Backbone	Configuration	mIoU	mASR
ResNet 50	Target only	65.2	100
	Source only	27.6	39.1
	MaxSquareIW [325] (r)	36.8	52.0
	UDA OCE [31] (r)	36.6	51.7
	LSR$^+$ (ours)	**40.9**	**58.6**
VGG 16	Target only	59.6	100
	Source only	25.5	42.4
	MaxSquareIW [325] (r)	31.7	46.9
	UDA OCE [31] (r)	34.2	51.5
	LSR$^+$ (ours)	**37.2**	**57.2**
VGG 13	Target only	59.5	100
	Source only	28.5	42.6
	MaxSquareIW [325] (r)	31.6	46.7
	UDA OCE [31] (r)	16.8	23.3
	LSR$^+$ (ours)	**36.3**	**55.5**

Table 9.9: Ablation Studies, mIoU and mASR scores comparison when removing any of the losses.

$\mathcal{L}_C$	$\mathcal{L}_P$	$\mathcal{L}_N$	$\mathcal{L}_{EM}$	mIoU	mASR
				42.8	64.4
	✓	✓	✓	44.9	66.3
✓		✓	✓	45.3	66.7
✓	✓		✓	46.0	68.3
✓	✓	✓		44.5	66.1
✓	✓	✓	✓	**46.9**	**69.5**

9.4.5.3 Results with Different Backbones

Table 9.8 shows the performance of our strategy on GTAV→Cityscapes using multiple encoder-decoder backbones in order to evaluate the generalization of the approach to different networks. Here we can see how LSR$^+$ outperforms the source-only models (*i.e.*, without adaptation) by 13.3%, 12.7% and 7.8% using ResNet50, VGG-16 and VGG-13, respectively. Even more importantly, we can see how the performance improvement is consistent across all backbones, in opposition to what happens to competing strategies. Finally, we remark the stability of the mASR score of our strategy, hovering around a mean of 57.0% with a very tight standard deviation of 1.4% (the other strategies have means 48.5% and 42.2%, and standard deviations of 2.45% and 31.3%, respectively).

9.4.6 Analyses of the Latent Space Regularization

In this section, we evaluate the impact of each component of the approach on the final accuracy. From Table 9.9 we appreciate that each component brings a significant improvement in terms of mIoU and that the best results are obtained when all components are enabled.

For visualization purposes, the plots of this section are computed on a balanced subset of feature vectors (250 vectors per class) extracted from the Cityscapes validation set for a fair comparative analysis across the classes.

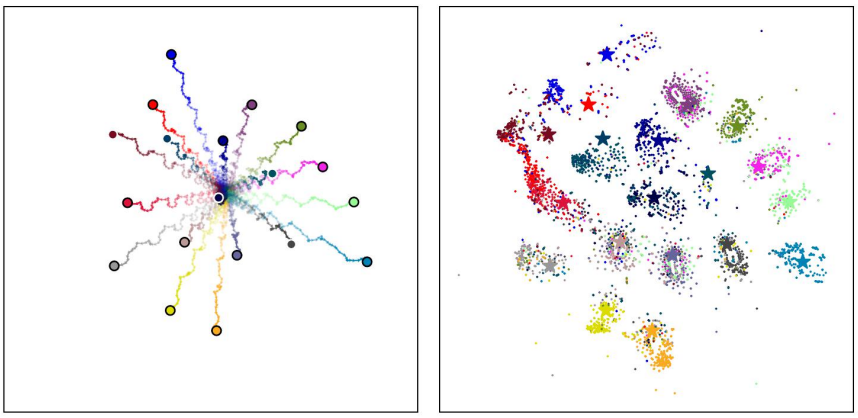
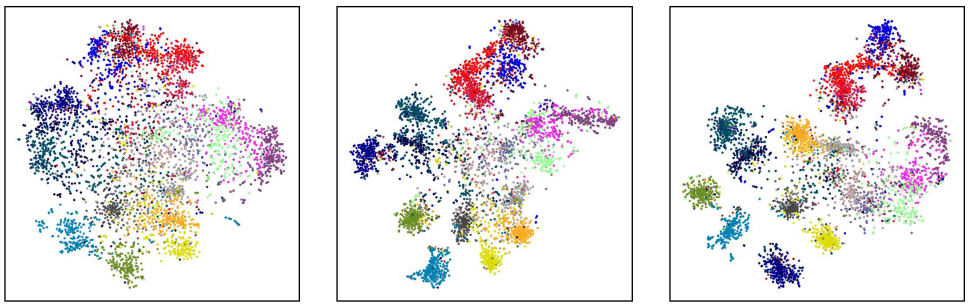


Figure 9.16: t-SNE embedding of the target feature vectors: trajectories of prototypes sampled over 200 training steps (on the left), features produced by the final model embedded according to the shared t-SNE projection (right).



(a) Baseline **(b)** LSR **(c)** LSR^{+}

Figure 9.17: t-SNE embeddings of the normalized feature vectors.

Two-pass prototypes and clustering. To investigate the semantic feature representation learning produced by our approach we computed a shared t-SNE [367] embedding on the prototypes sampled during the training procedure and of the target features produced by the final model. We remind the reader that, in order to more effectively shift target features closer to the source ones, we resort to a two-stage label assignment procedure which recovers target awareness (by averaging target-extracted features) from prototypes computed on the source domain (by centroid computation) as reported in Section 9.4.1-C. In the left plot of Figure 9.16 we report the learned prototype trajectory embeddings, and on the right the respective feature vectors. Here we can appreciate how prototypes get further apart while training goes on and how features extracted from the target domain lie in a neighborhood of the prototype, which we recall is computed exclusively via source-supervision. This underlines the effectiveness of our clustering strategy, which is able to shift the target feature distribution closer to the source one.

Finally, to further analyze our clustering objective we produce additional t-SNE embeddings starting from the normalized features (to remove the norm information, focusing on the angular one), which is reported in Figure 9.17. Our strategy increases significantly the cluster separation in the high dimensional space and the spacing between clusters belonging to different

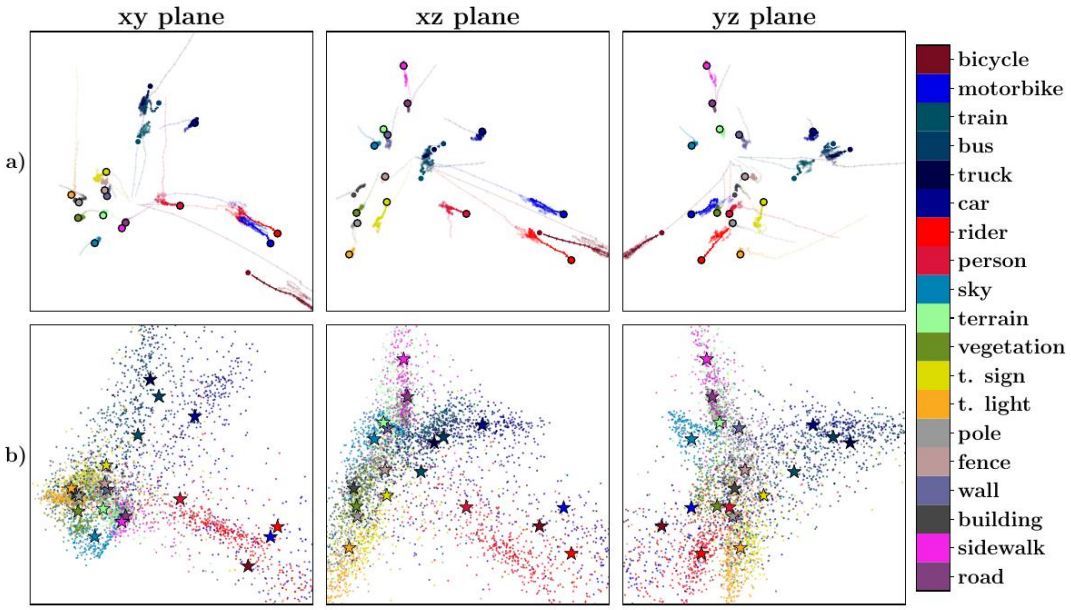


Figure 9.18: Prototypes trajectories and target feature vectors projected via PCA. Projection is 3-dimensional; here we report the three xy, xz and yz planes.

classes, promoting features disentanglement. As a side effect, this also reduces the probability of confusing visually similar classes (*e.g.*, the *truck* class with the *bus* and *train* ones).

In Figure 9.18 we report the PCA counterpart of Figure 9.17. Here we show how the distancing of the prototypes and source-target alignment is preserved even when projected using a linear function, as opposed to the non-linear t-SNE. In particular, Figure 9.18a) reports the prototypesà trajectories, while Figure 9.18b) reports the target vectors embedding.

Weighted histogram-aware downsampling. In this work, we extended the scheme proposed in [32] by adding class weights inversely proportional to the class-frequency in the training dataset (see Section 9.4.1). The goal of the proposed frequency-aware setup is to label only feature locations with a clear class assignment. This aims to reduce cross-talk between neighboring features of different classes, thus improving class discriminativeness at the latent space. We can observe this phenomenon in Figure 9.19, where the label map downsampled via our frequency-aware schemes (middle and right) marks some features close to the edges of objects as *unlabeled*, keeping only faithful features. As expected, class-weighting (right plot of Figure 9.19) promotes rarer classes at the feature level compared to the version without it [32] (middle plot of Figure 9.19): for instance, compare the traffic sign (in yellow). This is confirmed by the class distribution of the downsampled segmentation maps (*i.e.*, to match the spatial resolution of the latent space), reported in Figure 9.20 for our weighted histogram-aware scheme, the previous un-weighted histogram-aware scheme of the conference version [32] and the standard nearest neighbor. In particular, the schemes based on histogram-awareness generally seldom preserve small object classes, promoting *unlabeled* classification when discrimination between classes is uncertain. Our weighted histogram-aware scheme improves uniformity across rarer or smaller semantic categories, which were over-penalized by the previous approach [32], where all classes

(a) Nearest **(b)** Histogram **(c)** Weighted Histogram

Figure 9.19: Sample image downsampled nearest (left), frequency-aware [32] (middle) or weighted frequency-aware (LSR$^+$).

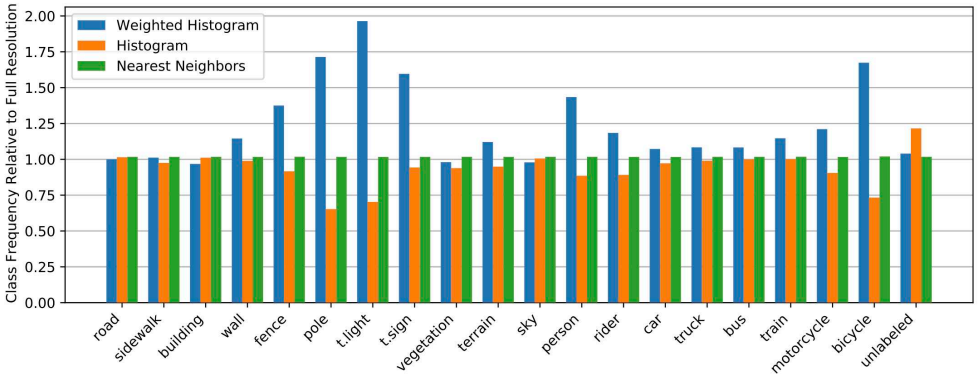


Figure 9.20: Class frequency of the downsampled feature-level segmentation maps.

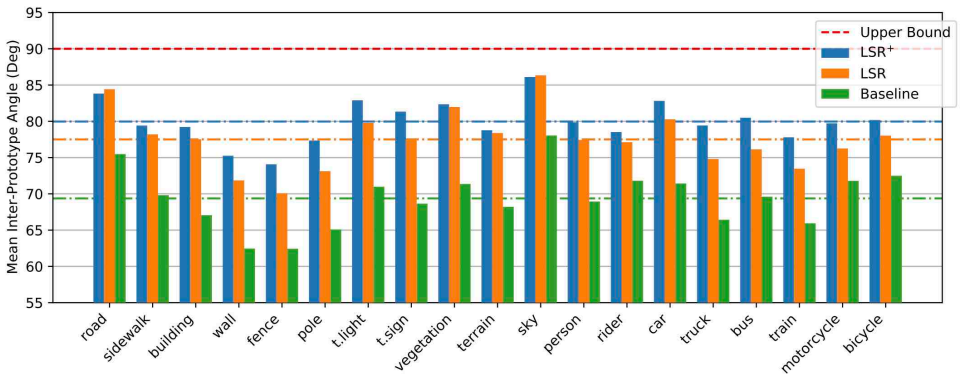


Figure 9.21: Average inter-prototype angle.

were treated equally, regardless of their occurrence.

Perpendicularity. To analyze the effect of the perpendicularity constraint, Figure 9.21 shows the distribution of the average inner angle between a prototype's direction and the direction of each other prototype. Ideally, we aim at producing as perpendicular prototypes as possible, in order to reduce the overlap of different semantic classes over feature channels (*i.e.*, cross-talk). The red dashed line at 90 degrees shows the target value for perpendicularity, which is also the upper bound for the angle, as our feature vectors have all non-negative coordinates. From the figure, it emerges clearly that LSR-based approaches increase the inter-prototypical

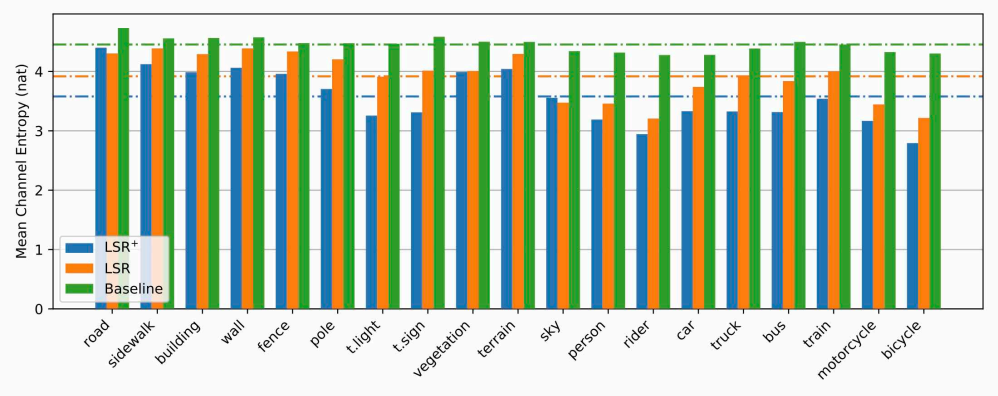


Figure 9.22: Average channel entropy.

angle and that LSR$^+$ makes prototypes even more orthogonal with an improvement of more than 2 degrees on average.

Norm Alignment. We analyze the effect of the norm alignment constraint in Figure 9.22, where we show the mean channel entropy for each class. We observe that the entropy corresponding to feature vectors produced by LSR$^+$ is significantly reduced, meaning that features are characterized by more relevant peaks and fewer poorly-activated channels.

9.5 Conclusions

In this chapter we discussed some novel schemes to perform unsupervised domain adaptation at the image level and at the feature level. In the first contribution we built a unified framework combining cycle-consistency and adversarial domain adaptation both at image level and at the feature level, differently from other competing works. We employed a MobileNet-v2 as segmentation network which is lightweight enough to allow a single shot end-to-end training on a single commercial GPU and to embed our full model in practical applications with limited computational constraints.

In the second contribution we proposed a novel feature-oriented UDA framework, comprising of three main objectives. First, features of same class and separate domains are clustered together, whilst features of different classes are spaced apart. Second, an orthogonality requirement over the latent space discourages the overlapping of active channels among feature vectors of different classes. Third, a sparsity constraint further reduces feature-wise the number of the active channels. All combined, these modules allow to reach a regularized disposition of latent embeddings, while providing a semantically consistent domain alignment over the feature space.

In the third contribution we introduced a set of latent-space regularization techniques to address the domain shift in an unsupervised fashion. We achieved domain invariance by means of multiple latent space-shaping constraints (namely, class clustering, class perpendicularity and norm alignment), to space apart features belonging to different classes while clustering together features of the same class in a consistent way on both the source and target domain. To support their computation, we introduced a novel target pseudo-labeling scheme and a weighted label decimation strategy. Finally, we designed a novel metric (mASR) to capture the relative performance between an adapted model and its target supervised counterpart.

We hope that our analyses on latent-level adaptation can pave the way to employment of

a new family of feature-level techniques to enhance the discrimination ability of deep neural networks; however, further research is needed to understand how to integrate our feature space adaptation with other approaches targeting different network levels.

9.6 Final Remarks

This second part of the dissertation on unsupervised domain adaptation in semantic segmentation started from the introduction of the problem and a careful categorization of the current literature on the topic in Chapter 7. We outlined three main level where adaptation can occur (namely at the output, input and feature level), and we proposed some techniques for each of the adaptation levels. We analyzed output-level UDA contributions in Chapter 8 by means of adversarial learning and self-training strategies. We presented input- and feature-level UDA contributions in Chapter 9. We employed CycleGAN to translate the input images across domains, and we applied feature-level disentanglement via adversarial learning, clustering, orthogonality and sparsity.

The combination of multiple levels of adaptation seems a viable path that future research could investigate to further improve the performance of an adapted deep neural model.

Part IV

Federated Learning of Visual Models

10
Federated Learning on Non-IID Data

This part of the dissertation moves away from the *classical* centralized training setup and explores Federated Learning (FL): a recently proposed training paradigm for decentralized model training. FL aims at training a machine learning algorithm, for instance deep neural networks, on multiple decentralized and private local datasets contained in local nodes without explicitly transmitting data samples. The general principle consists in training local models on local data samples and exchanging parameters (*e.g.*, the weights and biases of a deep neural network) between these local nodes at some frequency to generate a global model shared by all nodes. In this chapter, we aim at introducing the general FL setup and we propose a novel aggregation method FairAvg [34], which aggregates distributed contributions fairly from the user perspective (*i.e.*, each user is considered the same during aggregation), then we explore convergence properties and accuracy of aggregated models.

10.1 An Introduction to Federated Learning

FL systems enable distributed training of machine learning models in a network of clients (also called users, devices) with local data processed only at clients [371–374]. In FL systems, models are trained across multiple rounds. At the beginning of each round, every participating user receives an initial model from a central server, optimizes the model on its local training data and sends the updated model back to the server. The server then aggregates the received local solutions and updates the aggregate model [15]. Up to certain extents, FL is similar to distributed optimization in the context of machine learning, however, some key differences for what concern systems and statistical heterogeneity distinguish the two paradigms [15, 372, 373, 375]. The main difference between federated learning and distributed learning lies in the assumptions made on the properties of the local datasets, as distributed learning originally aims at parallelizing computing power where federated learning originally aims at training on heterogeneous datasets. While distributed learning also aims at training a single model on multiple servers, a common underlying assumption is that the local datasets are identically distributed (i.i.d.) and roughly have the same size. None of these hypotheses are made for federated learning; instead, the datasets are typically heterogeneous and their sizes may span several orders of magnitude. Moreover, the clients involved in federated learning may be unreliable as they are subject to more failures or drop out since they commonly rely on less powerful communication media (*i.e.*, Wi-Fi) and battery-powered systems (*i.e.*, smartphones and IoT devices) compared to distributed

learning where nodes are typically datacenters that have powerful computational capabilities and are connected to one another with fast networks [372, 376].

A major challenge for convergence of federated optimization is statistical heterogeneity. Whilst in centralized training data can be assumed i.i.d., decentralized data is generally highly imbalanced (*e.g.*, local data may contain different numbers of samples for different classes on each device) and non-i.i.d. (*e.g.*, samples in remote clients may have large correlation due to user-specific habits or preferences) [377].

Most of FL methods, starting from [15], aggregates weights of models according to local dataset sizes, implicitly claiming that models trained on more samples are *better and richer* compared to models trained with less samples, therefore adding more confidence to them. However, this policy misses other important properties of models and data, causing issues especially for training convergence. First, this leads to unfair aggregation with respect to users. Indeed, users with few local samples are considered less during aggregation and struggle to offer a real contribution to federated optimization of the models. Second, in real-world settings statistical heterogeneity (*e.g.*, highly imbalanced and non-i.i.d. data) is diffused and can seriously harm model training. Recently, some interest has been devoted to aggregation procedures and several attention methods employing functions of difference between parameters of local and aggregate models were proposed [378–382]. Nonetheless, also these methods fail in treating each user fairly, since users with few samples are considered less during aggregation and cannot bring a real contribution to federated optimization.

While it is known that FedAvg shows competitive results on i.i.d. data on convex loss landscapes [375, 383], it is clear that it cannot compete on non-i.i.d. and imbalanced data [383], as users with fewer samples (but potentially high statistical variability) are considered less during aggregation.

In this section, we show a comparative analysis of the baseline FedAvg, against a fair (uniform) aggregation scheme from the user perspective (FairAvg), to explore relationship between the two schemes in terms of convergence properties and accuracy of aggregated models. Experimental analyses over a suite of non-i.i.d. and imbalanced datasets show that a fair aggregation can be beneficial both for final accuracy and convergence rate, whilst at the same time reducing fluctuations of accuracy toward the convergence value (measured via the autocorrelation function). We observe that FairAvg is especially effective when few reporting clients participate in the aggregation and when each client sees few classes (non-i.i.d. split), since a few users with many local samples can bias the model toward the classes they observe if frequency-based aggregation is performed.

10.2 Problem Statement: FedAvg and FairAvg

In an FL system consisting of a set of clients $\mathcal{K} = \{1, 2, \ldots, K\}$, parameters $W_k \in \mathcal{W}_k$ of models $M_k : \mathcal{W}_k \times \mathcal{X}_k \to \mathcal{Y}_k$, are optimized at each client $k \in \mathcal{K}$ using its local dataset to learn feature representations, where $\mathcal{X}_k = \{\mathbf{x}_{k,j}\}_{j=1}^{n_k}$ and $\mathcal{Y}_k = \{\mathbf{y}_{k,j}\}_{j=1}^{n_k}$ denote respectively the set of samples and their ground truth labels (*e.g.*, one-hot encoded vectors of category labels for image classification, and vectors of segmentation maps for image segmentation) observed at the client k. In centralized FL systems, a central server coordinates the optimization of a set of parameters $\mathcal{W}$ of an aggregated model $M(\mathcal{W}, \cdot)$ by minimizing a global learning objective $L(W)$ [15] without sharing local datasets $\mathcal{S}_k = \{s_{k,j} = (\mathbf{x}_{k,j}, \mathbf{y}_{k,j})\}_{j=1}^{n_k}$ by solving

$$\min_{W \in \mathcal{W}} L(W) = \min_{W \in \mathcal{W}} \sum_{k \in \mathcal{K}} p_k L_k(W; \mathcal{S}_k), \tag{10.1}$$

where the local objective is computed by

$$L_k(W; \mathcal{S}_k) = \frac{1}{n_k} \sum_{j=1}^{n_k} l_k(W; s_{k,j} \in \mathcal{S}_k), \tag{10.2}$$

with $l_k(\cdot;\cdot)$ being a user-specific loss function, $p_k \geq 0$ is the weight of the objective $L_k(\cdot;\cdot)$ of the k^{th} client and $\sum_{k\in\mathcal{K}} p_k = 1$. McMahan *et al.* [15] proposed to use $p_k = \frac{n_k}{n}$, where $n = \sum_{k\in\mathcal{K}} n_k$. Thereby, $L(W)$ coincides with the training objective of the centralized setting. However, optimization of model parameters by minimization of the local objectives L_k for data distributions $\mathcal{D}_k$ over $\mathcal{X}_k \times \mathcal{Y}_k$ under different conditions (*e.g.*, imbalanced and non-i.i.d. data) is challenging hindering convergence of model parameters to optima.

Many FL systems have been designed to solve the problem (10.1). In our setup, we consider that first a subset $\mathcal{K}^t \subseteq \mathcal{K}$ of K' clients is randomly selected according to p_k at each federated round t. Then, selected clients download the aggregate model $W^t \in \mathcal{W}^t$ from a central server, perform local optimization minimizing an empirical objective $L_k(W^t; \mathcal{S}_k)$ with learning rate η for F epochs using a local optimizer such as SGD, and then send the final solution W_k^{t+1} back to the server. The server averages the solutions obtained from the clients with weights proportional to the size of the local datasets by

$$W^{t+1} = \sum_{k\in\mathcal{K}^t} \mathbf{a}^t[k] W_k^{t+1}, \tag{10.3}$$

where $\mathbf{a}^t$ is the *federated aggregation vector* at t which determines the importance of the received local models. The procedure is iterated for $T-1$ federated rounds and the final aggregate model is then identified by W^T.

Federated Averaging (FedAvg) is a benchmark federated optimization algorithm widely used to solve the problem (10.1). The popular federated optimizer FedAvg, widely used in FL systems, was proposed in [15]. FedAvg simply employs the frequency of local samples as federated aggregation vector, by setting

$$\mathbf{a}^t[k] = \frac{n_k}{\sum\limits_{j\in\mathcal{K}^t} n_j}, \quad \forall k \in \mathcal{K}^t, \forall t. \tag{10.4}$$

This choice was adopted by many recent methods [375,384–386], or replaced by attention values derived from statistical discrepancy measures of model weights [378–382].

Fair Averaging (FairAvg). While FedAvg prioritizes model weights according to the local frequency of samples, we argue that an unbiased fair policy for each user is to contribute equally to the aggregated model. Hence, we propose to define $\mathbf{a}^t$ by

$$\mathbf{a}^t[k] = \frac{1}{|\mathcal{K}^t|}, \quad \forall k \in \mathcal{K}^t, \forall t. \tag{10.5}$$

Fairness has been considered in resource division in multi-agent systems. A maximin sharing policy improves performance of the worst agent [387] and a fair-efficient policy makes variation of utilities of agents as small as possible [388]. Fairness in FL was examined from the perspective of ensuring accuracy across clients. Agnostic FL [389] minimizes the maximal loss function of all clients. In q-Fair FL [390], a more uniform accuracy distribution across clients is encouraged. In hierarchically fair FL [391], more contributions lead to more rewards. However, previous works ignore the fair user contribution. FedAvg and FairAvg approaches are summarized and compared

Algorithm 10.2 FedAvg and FairAvg.

Input: $\mathcal{K}, T, F, W^0, \eta, N$.
for $t = 0$ **to** $T - 1$
  A server samples $\mathcal{K}^t \subseteq \mathcal{K}$ clients and sends them W^t.
  for $k \in \mathcal{K}^t$
    Update W_k^t with L_k (10.2) and step size η to W_k^{t+1}.
    Send W_k^{t+1}back to the server.
  end for
  FedAvg. The server computes $\mathbf{a}^t$ via (10.4).
  FairAvg. The server computes $\mathbf{a}^t$ via (10.5).
  The server computes W^{t+1} via (10.3).
end for

Table 10.1: Statistics of the employed datasets (left) and hyper-parameters (right). In segmentation datasets, image background is excluded, and the accuracy refers to the mIoU. DeepLab-V3+ [4] uses MobileNet-v2 [360, 361] as the backbone pre-trained on ImageNet [330].

Dataset	# Classes	Clients	Samples	Samples/Client		Model	Distribution	Central. Acc. (%)	Start lr	Solver	F	Rounds	Batch size
				Mean	Std.								
Synthetic	10	30	9,600	320.0	1051.6	2 dense layers	Power-law	78.5	0.01	SGD	20	200	10
MNIST	10	1,000	61,676	61.7	164.7	2-layer CNN	Power-law	99.0	0.01	SGD	20	200	10
FEMNIST	10	200	16,421	82.1	143.0	2-layer CNN	Power-law	99.0	0.001	SGD	20	400	10
CelebA	2	9343	177,457	19.0	7.0	4-layer CNN	Power-law	92.6	0.1	SGD	20	200	10
FPascal Macro	4	100	6,665	66.7	25.7	DeepLab-V3+	Power-law	79.7	10^{-4}	Adam	2	400	16
FPascal	20	100	6,665	66.7	25.7	DeepLab-V3+	Power-law	66.3	10^{-4}	Adam	2	400	16
Sent140	2	772	40,783	53	32	Stacked-LSTM	Power-law	72.3	0.3	SGD	20	800	10
Shakespeare	80	143	517,106	3,616	6,808	Stacked-LSTM	Power-law	49.9	0.8	SGD	20	40	10

in Algorithm 10.2. Fairly treatment of users is a leading element in responsible AI [392].

10.3 Federated Learning Datasets

Some statistics of the employed federated datasets are reported in Table 10.1, inspired from [15, 393]. Synthetic data are sampled from a logistic regression model [375, 394]. MNIST [395] and FEMNIST [395] refer to image classification, FPascal to semantic segmentation, whilst Sent140 [396] and Shakespeare [397] to text-classification and next-character prediction, respectively.

10.3.1 Synthetic Data Classification Dataset

First of all, we analyse our approach on highly non-i.i.d. synthetic data. For this purpose, we follow a similar setup to that proposed in [375, 394] with the addition of heterogeneity among clients.

- **Synthetic:** For each client k, we generate samples $(\mathbf{x}_k, y_k) \in \mathcal{X}_k \times \mathcal{Y}_k$ according to the logistic regression model $y_k = \arg\max_{c \in \mathcal{C}}(\text{softmax}(W\mathbf{x}_k + b))$, $\mathbf{x}_k \in \mathbb{R}^{60}$, $W \in \mathbb{R}^{10 \times 60}$, $\mathbf{b}_k \in \mathbb{R}^{10}$. In the model, we first initialize $W_k \sim \mathcal{N}(u_k, 1)$, $\mathbf{b}_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, \phi_1)$, $\mathbf{x}_k \sim \mathcal{N}(\mathbf{v}_k, \mathbf{\Sigma})$, where the covariance matrix $\mathbf{\Sigma}$ is diagonal with $\mathbf{\Sigma}_{j,j} = j^{-1.2}$. Then, each element of $\mathbf{v}_k$ is drawn from $\mathcal{N}(B_k, 1)$, $B_k \sim \mathcal{N}(0, \phi_2)$. Hence, ϕ_1 controls how much local models differ from each other, ϕ_2 controls how much local data distribution at each client differs from that of other clients. For our simulations, we set $\phi_1 = \phi_2 = 1$ being the most heterogeneous, yet challenging, scenario. Other analyses have been carried out in [375].

Assuming that the underlying data generation model is agnostic, we employ a cascade of 2 dense layers with 128 and 256 units respectively, followed by a softmax output layer.

10.3.2 Real-World Image Classification Datasets

To further investigate accuracy on classification data, we explore real-world image classification datasets, inspired from [15, 375, 393].

- **MNIST:** It is a classification task of $28 \times 28px$ images containing handwritten digits 0-9 [395]. To simulate a heterogeneous (non-i.i.d.) setting, we distribute data among $1,000$ clients such that each client has samples of only two digits and the number of samples per client follow a power law distribution. To tackle this task we employ a simple custom network with 5×5 convolution layers (the first with 32 channels, the second with 64, each followed by 2×2 max pooling), a fully connected layer with 256 units and ReLu activation, and a final softmax output layer.
- **FEMNIST:** It is a classification task of $28 \times 28px$ images containing 62-class handwritten character digits [395]. Following [375], to generate heterogeneity we first subsample 10 lower case characters (from 'a' to 'j') and we distribute only 5 classes to each client. The number of clients is 200. To tackle this task we employ a simple custom network with 5×5 convolution layers (the first with 32 channels, the second with 64, each followed by 2×2 max pooling), a fully connected layer with 256 units and ReLu activation, and a final softmax output layer.
- **CelebA:** Finally, we generate non-i.i.d. CelebA [398] data (for classification of smiling faces), such that the underlying distribution of data for each user is consistent with the raw data. For this task, we use a 4-layer CNN each with 32 channels and followed by 2×2 max pooling and ReLU activation, and a softmax layer.

10.3.3 Real-World Semantic Segmentation Datasets

Then, we investigate the accuracy on a dense prediction task, such as semantic segmentation. We propose a new benchmark for federated semantic segmentation employing the Pascal VOC2012 dataset [114] in two different flavours.

- **Pascal:** We use the standard Pascal VOC2012 [114] semantic segmentation dataset. We consider only images with one single class inside (in addition to the background) in order to mimic classification splits. There are a total of 20 object-level classes (background excluded) and we distribute data to each client according to a Dirichlet distribution with concentration parameter $\alpha > 0$. Low values of α mean that dataset is highly non-i.i.d. among clients, and vice-versa for high α values [385,399]. The number of samples per client follow a power-law distribution with parameter $\gamma = 3$. For this task, we use a DeepLab-V3+ [4] architecture with MobileNet [361] as the encoder pre-trained on ImageNet [330]. To further elucidate on the data splitting mechanism, we report some dataset statistics for different values of α in Figure 10.1. The first column represents the distribution of the number of classes present on clients, while the second column represents the distribution of clients having a certain amount of classes. In the most non-i.i.d. case considered (*i.e.*, $\alpha = 0.01$), each client only experiences samples from a few classes, while they progressively observe more and more classes as the i.i.d.-ness improves. The third column of Figure 10.1 illustrates populations drawn from the Dirichlet distribution with different concentration

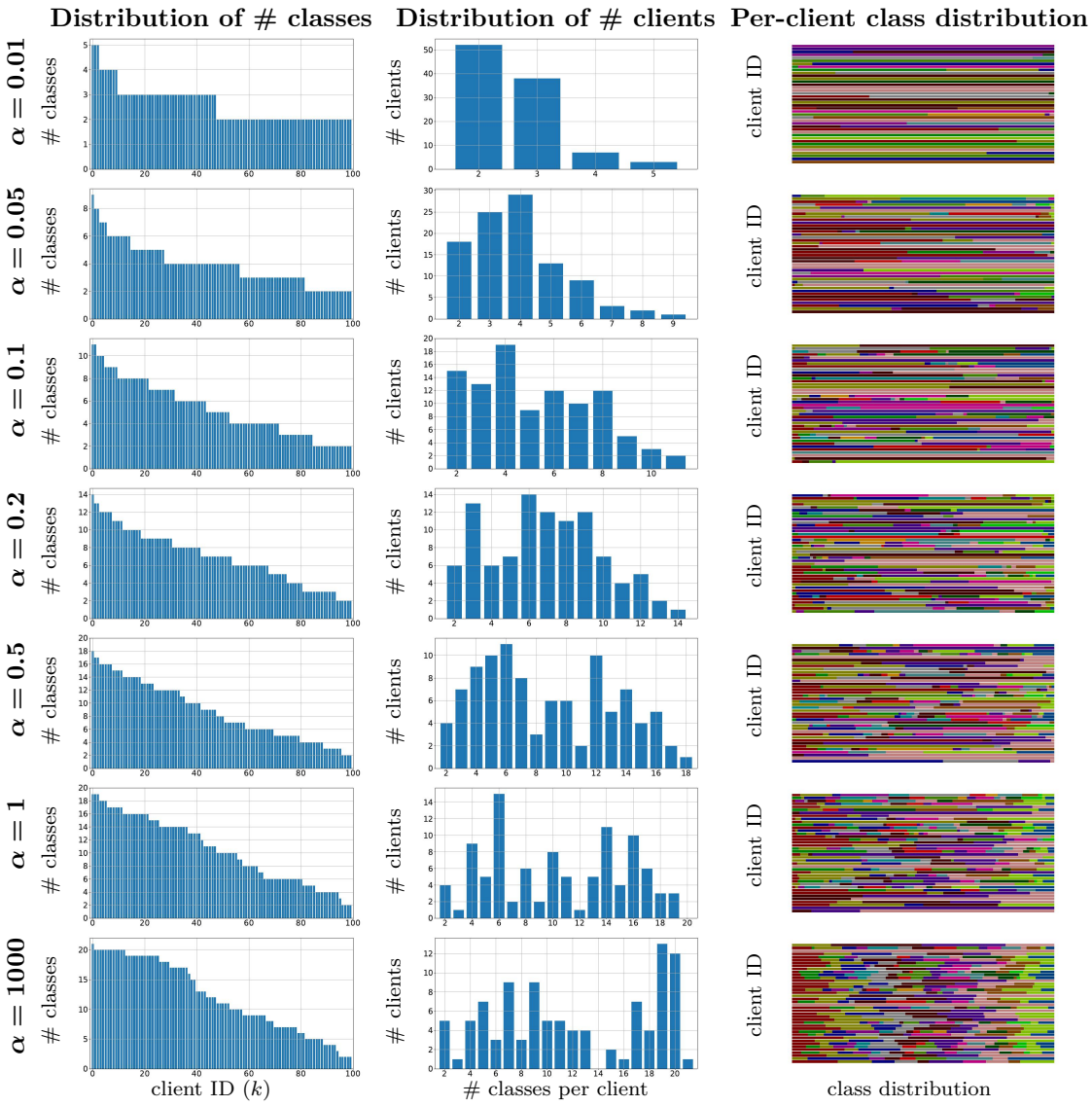


Figure 10.1: Dataset statistics of different data splitting schemes used by clients for the Pascal VOC2012 segmentation task. The first column reports the distribution of the number of classes among clients (note that the background is present in all the images). The second column shows the distribution of number of clients according to number of classes per client. The third column reports the per-client distribution of classes depicted with different colors, where the client IDs are restricted to 30 randomly sampled clients for visualization purposes (the background is not included in the visualization and the colors refer to the Pascal VOC2012 colormap).

parameters. For visualization purposes, we restrict to 30 randomly sampled clients and each color refers to a different class (color coding scheme reflects Pascal VOC2012 colormap). As expected, for low values of α, the distributions are similar but not identical to a sort-and-partition approach (clients represented by very few colored segments) in which each client only sees samples of one class (plus the background) [399], since we have a highly imbalanced number of samples per each class and samples are distributed to clients

according to a power-law distribution. Experimentally, we verified that a simple extremely i.i.d. sort-and-partition approach yields same results as the case with $\alpha = 0.01$. For high values of *alpha*, instead, the distribution of classes visible at clients becomes more i.i.d. as each client sees samples from may different classes (clients represented by many segments of different colors).

- **Pascal macro:** We follow the same exact splitting described for standard Pascal VOC2012 with the only difference that classes are hierarchically grouped according to their semantic meaning into 5 classes (background included). The coarser set of classes is derived from the notional taxonomy from [114, 400]. The map from 21 to 5 classes is:
  - Background: *Background*;
  - Person: *Person*;
  - Vehicles: *Aeroplane*, *Bicycle*, *Boat*, *Bus*, *Car*, *Motorbike*, Train;
  - Household: *Bottle*, *Chair*, *Dining Table*, *Potted Plant*, *Sofa*, *TV/Monitor*;
  - Animals: *Bird*, *Cat*, *Cow*, *Dog*, *Horse*, *Sheep*.

10.3.4 NLP Datasets

Finally, for some analyses we considered two NLP classification datasets to demonstrate the generalization capability of our proposed methods to other tasks and domains.

- **Shakespeare:** it is a language modeling dataset extracted from *The Complete Works of William Shakespeare* [15, 397]. Each speaking role in each play with at least two lines represent a different client. The task is next-character prediction. This produced a dataset with 1146 clients and there are 80 characters in total. To tackle this task, we used a two-layer stacked LSTM with 100 hidden units followed by a densely-connected layer. We set the length of the input sequence to 80 characters and we embed each character into a learned 8-dimensional space.
- **Sent140:** it is a sentiment analysis task on tweets from Sentiment140 [396], where each Twitter account corresponds to a client. To tackle this task, we used a two-layer stacked LSTM binary classifier with 256 hidden units and pretrained 300D GloVe embedding [401] followed by a densely-connected layer. We set the length of the input sequence to 25 characters and we embed each character into a 300-dimensional space.

10.4 Experimental Evaluation of Data Non-IID-ness

In this section, we investigate the relationship between statistical properties of federated aggregation vectors $\mathbf{a}^t, \forall t$, distribution of number of classes among clients, accuracy of models and their stationarity over aggregation rounds.

Figure 10.2 shows the per-round aggregate accuracy (original accuracy values in row 1 and values smoothed over a window of 10% rounds for visualization in row 2) and the training loss (original in row 3 and smoothed in row 4). Further analyses of accuracy of aggregate models achieved at the final round are given in Table 10.2 for a different number of reporting clients K'. In these results, FairAvg demonstrates to be particularly effective when as few as $K' = 2$ reporting clients are considered and to outperform FedAvg overall.

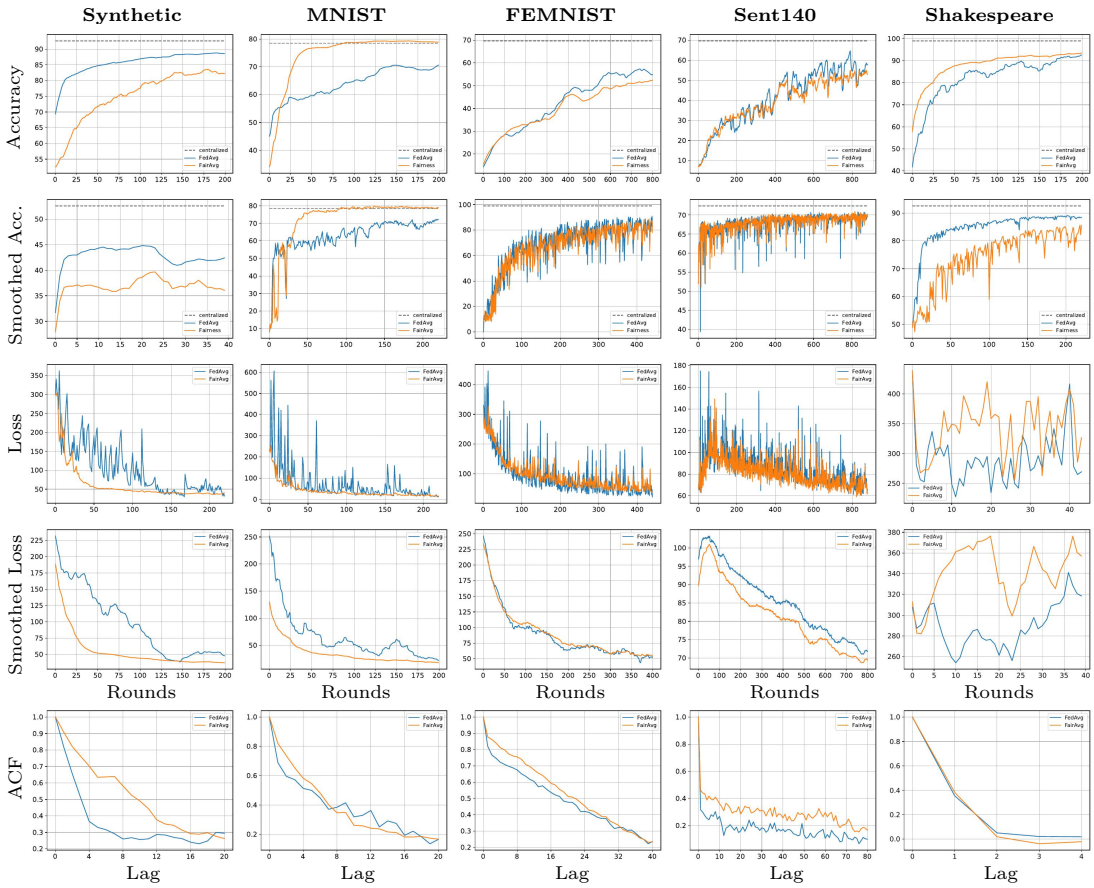


Figure 10.2: Classification accuracy (%), training loss and their respective smoothed versions over a window of 10% rounds (which are smoothed for visualization). Last row reports the correlogram of the accuracy (reported as first row), *i.e.* a plot of the autocorrelation function (ACF) for sequential values of lag. Different datasets are considered over the columns and $K' = 10$ reporting users.

From first row of Figure 10.2, we observe that a fair policy is especially helpful on synthetic and MNIST datasets, where it robustly outperforms FedAvg by a large margin. No clear winner emerges on FEMNIST and Sent140 datasets, and FedAvg surpasses FairAvg on Shakespeare data.

Table 10.2: Accuracy (%) of the aggregate model on the final round for different number of reporting clients K'.

	Synthetic			**MNIST**			**FEMNIST**			**Sent140**			**Shakespeare**		
K'	2	5	10	2	5	10	2	5	10	2	5	10	2	5	10
FedAvg	75.9	70.4	70.6	85.1	88.8	92.6	66.2	77.0	**82.3**	63.3	67.2	**69.5**	**36.1**	**41.1**	**42.4**
FairAvg	**76.4**	**78.7**	**78.9**	**86.5**	**89.9**	**92.8**	**70.9**	**77.6**	81.5	**63.6**	**67.5**	69.3	**36.1**	40.5	42.1

Beside the improvement in terms of accuracy, we remark how the fairness policy shows much faster convergence and higher training stability. We can observe this latter claim by visually inspecting the amount of fluctuations of accuracy (related to stationarity). While FedAvg (blue curve) shows many bursts and irregular peaks and pitfalls, FairAvg (orange curve) generally shows a more smoothed path toward the convergence value. This is due to the tailed distribution

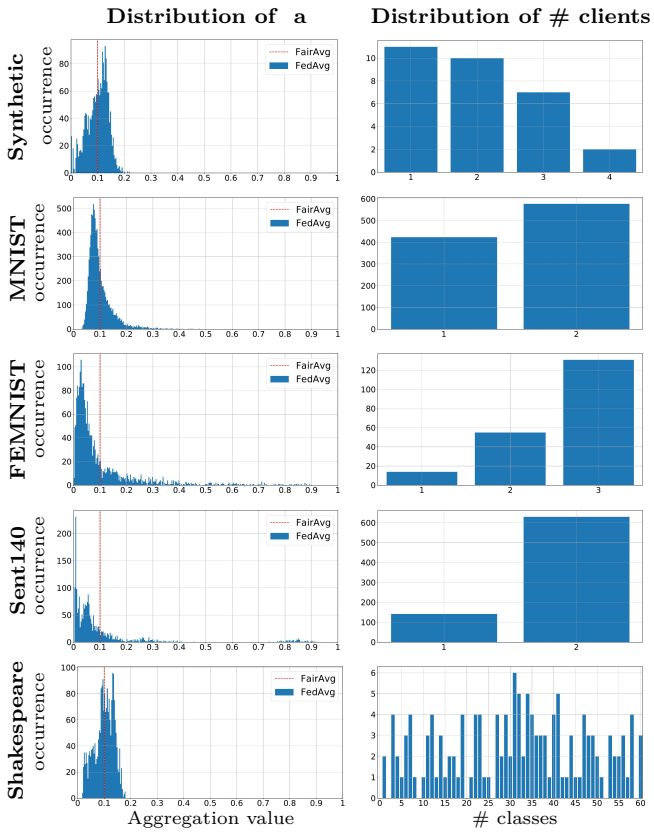


Figure 10.3: Distribution of federated aggregation values $\mathbf{a}^t[k], \forall k, \forall t$ (left) and distribution of number of classes into clients (right), over different datasets for $K' = 10$ reporting users.

of local samples on non-i.i.d. and imbalanced datasets. We quantitatively measure stationarity computing the correlogram, *i.e.*, a plot of the autocorrelation function (ACF) for sequential values of lag. Given a time-series accuracy vector $\mathbf{f}^t$ (*i.e.*, the series of accuracy values computed at each round and reported in first row of Figure 10.2), its autocorrelation function at lag l is defined by

$$ACF(\mathbf{r}, l) \triangleq \frac{\mathbf{r}[l]}{\mathbf{r}[0]}, \quad \forall l, \tag{10.6}$$

where $\mathbf{r}[l]$ represents the vector computed using the sample autocovariance function for lag l defined by

$$\mathbf{r}[l] = \frac{1}{T}\sum_{t=1}^{T-l}(\mathbf{f}^t - \bar{f})(\mathbf{f}^{t+l} - \bar{f}), \tag{10.7}$$

where $\bar{f} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{f}^t$ denotes the average value.

The correlogram, then, shows stationarity of the time series or change of fluctuations in the convergence of model parameters during federated optimization. Higher values of ACF denote lower fluctuations, which are adverse to a smoothed convergence of the aggregate model parameters to the final accuracy. From the plots, we observe that FairAvg robustly shows much higher ACF values and thus lower fluctuations while reaching the final accuracy value.

Distribution of federated aggregation values $\mathbf{a}^t[k], \forall k, \forall t$ used by FedAvg is reported in the left column of Figure 10.3 against the value employed by FairAvg. We remark that distribution of aggregation values computed by FedAvg reflects information on the distribution of local number of samples across clients by definition. More precisely, we observe tailed distributions (as a direct consequence of power-law data splitting over the clients) where a large number of users have fewer local samples compared to the case where datasets are distributed following a balanced splitting scheme. Thereby, model parameters of most users are weighted by lower federated aggregation values while aggregating models using FedAvg, compared to their aggregation by the FairAvg scheme. As a result, a small number of users with many local data tend to influence more, and eventually dominate, the resulting aggregated model.
This can be further verified in the right column of Figure 10.3, showing the distribution of clients having a certain amount of classes within their local data. In particular, we notice that the total number of classes for Synthetic, MNIST, FEMNIST, Sent140 and Shakespeare datasets is 10, 10, 10, 2 and 80, respectively (see Table 10.1). In the reported plots, instead, we can visualize how each device observes much less classes in its local samples, thus hindering optimal convergence. On synthetic data, users only see up to 4 classes (*i.e.*, 40% of the total number of classes), on MNIST up to 2 (*i.e.*, 20%), on FEMNIST up to 3 (*i.e.*, 30%), on Sent140 up to 2 (*i.e.*, 100%) and on Shakespeare up to 61 (*i.e.*, 76.3%). Excluding binary classification on Sent140, where results are overall even, then we observe that FairAvg approach outperforms FedAvg especially when each client sees a lower percentage of number of classes (*e.g.*, on Synthetic and MNIST datasets), while it is surpassed when each client observes a higher percentage (*e.g.*, on Shakespeare data).

10.5 Summary

Popularly employed federated optimization methods, such as FedAvg, aggregate models of users with importance proportional to the frequency of their local samples. However, this leads to unfair aggregation with respect to users.

In this section, we proposed a set of experiments to empirically explore the relationship between fairness of aggregation schemes, accuracy of aggregated models and convergence rate of federated optimization methods.

Experimental results on non-i.i.d. data showed that a fair aggregation scheme is beneficial compared to FedAvg for both final accuracy and convergence rate, whilst reducing at the same time fluctuations of accuracy of the aggregate model. Following experimental evidence, we believe that FL models could employ federated aggregation values centered around the value employed by FairAvg for uniform treatment of user contributions, as it is described in the next chapter.

11

Federated Learning of Visual Feature Representations

As we observed in Chapter 10, Federated Learning (FL) is a framework which enables distributed model training using a large corpus of decentralized training data. Existing methods aggregate models disregarding their internal representations, which are crucial for training models in vision tasks. System and statistical heterogeneity (*e.g.*, highly imbalanced and non-i.i.d. data) further harm model training. To this end, we introduce a method, called FedProto, which computes client deviations using margins of prototypical representations learned on distributed data, and applies them to drive federated optimization via an attention mechanism [35]. Thus, FedProto brings class-conditional prototype awareness to clients, differently from the competitors. In addition, we propose three methods to analyse statistical properties of feature representations learned in FL, in order to elucidate the relationship between accuracy, margins and feature discrepancy of FL models. In experimental analyses, FedProto demonstrates competitive accuracy and convergence rate across image classification and semantic segmentation benchmarks by enabling maximum margin training of FL models. Moreover, FedProto reduces uncertainty of predictions of FL models compared to the baseline. To our knowledge, this is the first work evaluating FL models in dense prediction tasks, such as semantic segmentation.

11.1 Introduction

Challenges of FL: Training models in FL systems introduces several novel challenges [372,373]. In this work, we address problems caused by system and statistical heterogeneity. System heterogeneity refers to variable computational (*e.g.*, CPU, memory, battery level) and communication (*e.g.*, wifi) capabilities of each device [402, 403]. Early approaches suggest to drop devices that fail to compute pre-determined workloads within a time window [15,371]. However, Li *et al.* [375] showed that this has negative effects on convergence as it limits the number of effective devices contributing to training and may induce bias, if dropped devices have specific data characteristics. Hence, we tolerate partial workload on clients following recent works [375, 404].

Statistical heterogeneity reflects another major challenge for convergence: whilst in centralized training, data can be assumed independent and identically distributed (i.i.d.), decentralized data is generally highly imbalanced (*e.g.*, local data may contain different number of samples for different classes on each device) and non-i.i.d. (*e.g.*, samples in remote clients may have large

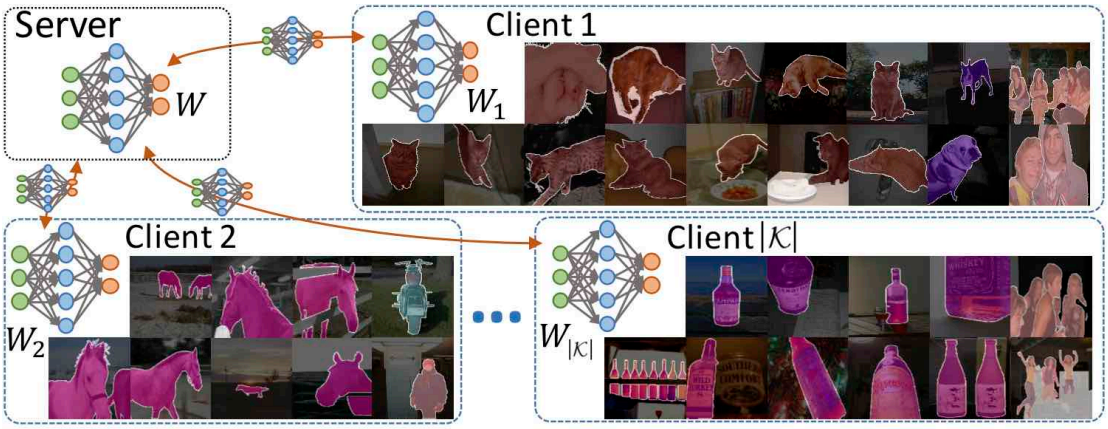


Figure 11.1: Visual data observed at distributed clients $k \in \mathcal{K}$ are non-i.i.d. and imbalanced. This represents a challenge for federated learning of vision models with parameters $W_k, \forall k$.

correlation due to user-specific habits or preferences) [377], as depicted in Figure 11.1.

Challenges of FL of visual feature representations: Representation learning has been a prosperous technique used to perform complex computer vision tasks, such as image classification and segmentation [102, 103]. In this paradigm, a model is trained to learn *rich* feature representations of its inputs, and learned representations are employed by task specific predictors (*e.g.*, classifiers or detectors). Current FL approaches focus on learning features by considering only statistical properties of data, such as joint distribution of samples and their class labels [385], and weights of models [375]. In FedAvg [15], weights are aggregated with importance scores proportional to size of local datasets, ignoring the learning dynamics. A similar approach has been followed by many subsequent methods [375,384–386]. More recently, increasing interest has been devoted toward elucidating aggregation procedures. Attention methods [378–382] were proposed using functions of difference between parameters of local and aggregate models. However, these works disregard relationship between statistical properties of the learned representations.

Here, instead, we propose a prototype guided federated optimization method (FedProto), which leverages the model aggregation procedure by prioritizing distributed models on basis of their learned prototypical representations of object categories. In particular, FedProto consists of three steps:

(i) Prototypical representations: First, we compute prototypical representations using local and aggregate models motivated by their success in meta-learning [149, 405–409], domain adaptation [13, 31, 410], semantic segmentation [140, 141] and continual learning [21, 411].

(ii) Confidence of local and aggregate models: Second, we compute confidence of local and aggregate models with respect to their decision on local data using prototypical (hypothesis) margins (PMs). PMs have been explored for developing learning vector quantization (LVQ) methods [412–416]. In [417], PMs are shown to lower bound sample margins and provide a rigorous upper bound of generalization error. Unlike PMs proposed for individual models, we aim to measure the change in semantic representations of FL models learned at different clients and over different rounds considering their generalization properties. Therefore, we first define a novel semantic PM. Then, motivated by these theoretical results, we drive the model aggregation process combining a confidence measure computed between local prototypes at the beginning and at the end of the local optimization (*i.e.*, *Local PM*), as well as a measure computed between aggregate and local prototypes at the server-level (*i.e.*, *Aggregate PM*). Although margins

between features and prototypes have been used to solve other vision tasks (*e.g.*, few-shot learning [408]), to our knowledge, our work is the first to compute margins among sets of prototypes and employ them for federated optimization.

(iii) Prototype-based weight attention: Finally, we propose a weight attention mechanism during global aggregation of local models using non-linear functions (*e.g.*, sigmoid) of prototypical margins. In FL, state-of-the-art attention methods [379, 381] consider only statistics of local models ignoring their effect on decision boundary. Instead, our attention mechanism quantifies this information by margins and employs it for aggregation. We conjecture that our mechanism enables maximization of latent-level margins in FL, which is experimentally justified in Section 11.5.

Intuitively, driving the model to focus on class prototypes, we achieve a better shaping of the inner space (thus acting as regularization constraint) that eventually eases the classifier task, which is a harder task than feature extraction [418]. Therefore, FedProto shows better convergence rate and accuracy, ultimately achieving a closer latent space organization to the one centralized training would produce.
The main contributions of our work are as follows:

- We propose a novel FL algorithm (FedProto) which applies a *prototypical margin-based model attention mechanism* to drive FL optimization in heterogeneous systems.
- We achieve state-of-the-art results on a variety of image classification and semantic segmentation benchmarks. To the best of our knowledge, this is the first work exploring federated learning of semantic segmentation models.
- We propose two quantitative metrics and a qualitative method based on entropy maps to analyse statistical properties of feature representations learned in FL systems.

11.2 Prototype Guided Federated Learning

Prototypical representations have been successfully employed in various computer vision tasks [21, 149, 407, 410]. In this work, we employ prototypes for federated optimization of vision models. Our prototype guided federated optimizer (FedProto) is motivated by the results obtained from the recent theoretical and experimental analyses of generalization capacity of latent class-conditional prototypes [149, 407].

Partial workload toleration: Optimizing local models with the same number of local epochs at all clients is unfeasible for real-world applications [371, 375, 385]. A more natural approach is to allow the epochs to vary according to the characteristics of the FL system, and to properly merge solutions accounting for heterogeneity of the system. Indeed, we observe that different clients in FL systems are likely to have very different resource constraints (causing system heterogeneity), such as different data resources, hardware configurations (*e.g.*, visual sensors, cameras or processors), network connections and battery levels [372]. Therefore, we allow partial amount of work to be conducted locally by each client prior to the aggregation stage, as utilized in FedProx [375]. In other words, at each round, instead of dropping $\delta\%$ of clients that performed less epochs than the total number F in a predetermined amount of time, we aggregate all the solutions sent from local clients tolerating partial workload, *i.e.*, even if the completed number of local epochs is $F' < F$. Following [375], we mimic this behavior by uniformly sampling $F' \sim \mathcal{U}([0, F))$ on each client.

At each round t and client k, a local model $M_k^t(\mathcal{W}_k^t; \mathcal{X}_k) = C_k^t(\mathcal{W}_{c,k}^t) \circ E_k^t(\mathcal{W}_{e,k}^t; \mathcal{X}_k)$ is computed, where $\circ$ denotes function composition, and $\mathcal{W}_{c,k}^t \subset \mathcal{W}_k^t$ and $\mathcal{W}_{e,k}^t \subset \mathcal{W}_k^t$ denotes sets of parameters of classifiers and encoders embodied in the model M_k^t, respectively. For each input $\mathbf{x}_{k,j} \in \mathcal{X}_k$, its latent representation $\mathbf{e}_{k,j}^t = E_k^t(\mathcal{W}_{e,k}^t; \mathbf{x}_{k,j})$ is computed and then fed to a classifier

$C_k^t(\mathcal{W}_{c,k}^t; \mathbf{e}_{k,j}^t)$ to retrieve class-wise probability scores. Features corresponding to the same class are then averaged to construct local latent class-conditional prototypes.

11.2.1 Computation of Prototypes

At each round $t > 0$, class $c \in \mathcal{C}$ and client $k \in \mathcal{K}^t$, the c^{th} element of prototypes $\mathbf{p}_k^t$ is computed by

$$\mathbf{p}_k^t[c] = \sum_{\mathbf{e}_{k,j,c}^t \in \mathcal{F}_{k,c}^t} \frac{\mathbf{e}_{k,j,c}^t}{\mathbf{n}_k^t[c]}, \quad \forall c \in \mathcal{C}, \forall k \in \mathcal{K}^t, \tag{11.1}$$

where $\mathcal{F}_{k,c}^t$ is the set of feature vectors $\mathbf{e}_{k,j,c}^t$ extracted from the sample $\mathbf{x}_{k,j} \in \mathcal{X}_k$ belonging to the class c, and $\mathbf{n}_k^t[c] = |\mathcal{F}_{k,c}^t|$ is the cardinality of $\mathcal{F}_{k,c}^t$. At $t = 0$, we initialize prototypes as $\mathbf{p}_k^0[c] = \mathbf{0}, \forall c, k$. Since features representing different classes have variable norm [419], we employ min-max normalization over the channels and denote the normalized prototypes by $\hat{\mathbf{p}}_k^t$.

11.2.2 Local and Aggregate Prototype Margins

To guide the optimization, we rely on a combination of two clues derived from displacement of prototypes:

1. *Local Prototype Margin (LPM)* measures deviation of on-client prototypes before and after local training.
2. *Aggregate Prototype Margin (APM)* measures deviation of aggregate prototypes from local prototypes.

As a measure for displacement, we embraced the margin theory [408, 412, 414, 415, 417, 420], in which PMs measure the distance between features and class decision boundaries. In our work, instead, we aim to measure change of semantic representations among clients over different rounds for FL. Therefore, we propose a novel semantic PM next.

Definition 11.2.1 (Semantic PM - SPM)**.** Given two prototype vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ defined on the same class space $\mathcal{C}$, we restrict to $\mathcal{C}' \subset \mathcal{C}$ such that $\mathbf{n}_i[c] > 0$ and $\mathbf{n}_j[c] > 0$, $\forall c \in \mathcal{C}'$, the distance between prototypes corresponding to the same semantic label c is computed by

$$\mathbf{d}_{i,j}^+[c] = d(\mathbf{p}_i[c], \mathbf{p}_j[c]), \quad \forall c \in \mathcal{C}', \tag{11.2}$$

and the average distance between prototype of a certain class c and prototypes of different classes is computed by

$$\mathbf{d}_{i,j}^-[c] = \sum_{\substack{c' \neq c \\ c' \in \mathcal{C}'}} \frac{d(\mathbf{p}_i[c], \mathbf{p}_j[c'])}{|c' \neq c \wedge c' \in \mathcal{C}'|}, \quad \forall c \in \mathcal{C}'. \tag{11.3}$$

Then, the SPM for class c is defined by

$$\mu(\mathbf{p}_i[c], \mathbf{p}_j) \stackrel{def}{=} \frac{\mathbf{d}_{i,j}^-[c] - \mathbf{d}_{i,j}^+[c]}{\mathbf{d}_{i,j}^-[c] + \mathbf{d}_{i,j}^+[c]}, \quad \forall c \in \mathcal{C}'. \tag{11.4}$$

In FL, we employ SPMs in two cases, LPM and APM, which are defined in Definitions 11.2.2 and 11.2.3. In the analyses, we identify $d(\cdot, \cdot)$ by the Euclidean distance, since it has been shown to outperform cosine similarity [140, 149] or to achieve comparable performance [141, 150].

Definition 11.2.2 (LPM)**.** The LPM is defined by

$$\boldsymbol{\mu}_{\text{loc},k}^t[c] \stackrel{def}{=} \mu(\hat{\mathbf{p}}_k^{t-1}[c], \hat{\mathbf{p}}_k^t), \quad \forall k \in \mathcal{K}^t, \forall c \in \mathcal{C} \tag{11.5}$$

and it measures change of local prototypes obtained from local models before and after their local training.

Definition 11.2.3 (APM)**.** The APM is defined to measure discrepancy between local and aggregate set of prototypes by

$$\boldsymbol{\mu}^t_{\text{agg},k}[c] \stackrel{def}{=} \mu(\hat{\mathbf{p}}^t_k[c], \hat{\mathbf{p}}^{t-1}_{\text{agg}}), \quad \forall k \in \mathcal{K}^t, \forall c \in \mathcal{C} \tag{11.6}$$

where the aggregate set of prototypes is defined by

$$\hat{\mathbf{p}}^t_{\text{agg}}[c] \stackrel{def}{=} \sum_{k \in \mathcal{K}^t} \frac{\mathbf{n}^t_k[c]}{\mathbf{n}^t_{\text{agg}}[c]} \hat{\mathbf{p}}^t_k[c], \quad \forall c \in \mathcal{C}, \tag{11.7}$$

with aggregate number of features $\mathbf{n}^t_{\text{agg}}[c] = \sum_{k \in \mathcal{K}^t} \mathbf{n}^t_k[c]$, $\forall c \in \mathcal{C}$ and initialization $\hat{\mathbf{p}}^0_{agg}[c] = \mathbf{0}$.

We remark that APM requires transmission of prototypes from clients to server. However, this does not raise privacy issues since prototypes represent only an averaged statistic over all local data of already compressed feature representations, nor large communication overhead, as the size of prototypes is negligible compared to the model size. Additionally, prototypes can be shared with *differential privacy* without inadvertently leaking private information [421, 422].

While local deviation measured by LPM gives a hint of how much a model adapts its inner representation for each class, server-side deviation measured by APM tells how much a local model changes its inner representations with respect to the prototypical representations aggregated over previous rounds and clients. The effect of distributed versus centralized calculation of margins is analysed in Section 11.5.1.

11.2.3 Federated Attention using Prototype Margins

Client deviations are computed by summing over all the classes and applying a sigmoid function σ [414, 415] by

$$\mathbf{v}^t_\iota[k] = \sigma\left(\sum_{c \in \mathcal{C}} \boldsymbol{\mu}^t_{\iota,k}[c]\right), \quad \forall k \in \mathcal{K}^t, \iota \in \{\text{loc}, \text{ agg}\}. \tag{11.8}$$

Definition 11.2.4 (Local, aggregate and federated attention)**.** A local (aggregate) weight attention vector $\mathbf{a}^t_{\text{loc}}$ ($\mathbf{a}^t_{\text{agg}}$) is computed normalizing the client deviations by

$$\mathbf{a}^t_\iota[k] \stackrel{def}{=} \frac{\mathbf{v}^t_\iota[k]}{\sum_{j \in \mathcal{K}^t} \mathbf{v}^t_\iota[j]}, \quad \forall k \in \mathcal{K}^t, \iota \in \{\text{loc}, \text{ agg}\}. \tag{11.9}$$

The federated weight attention vector $\mathbf{a}^t$ is defined by

$$\mathbf{a}^t[k] \stackrel{def}{=} \begin{cases} \frac{n_k}{\sum_{j \in \mathcal{K}^t} n_j}, & \text{if } t = 0 \\ \frac{\mathbf{a}^t_{\text{agg}}[k] + \mathbf{a}^t_{\text{loc}}[k]}{2}, & \text{if } t > 0 \end{cases} \tag{11.10}$$

Intuitively, each $\mathbf{a}^t[k]$ represents a measure of client drift: as prototypes computed using weights $W_k \in \mathcal{W}_k$ of a model of a client k deviate from reference prototypes in terms of margin (either locally or on server), higher attention is applied on the weights $W_k \in \mathcal{W}_k$, and vice-versa. We remark that, according to our definition, if a client is not able to build reliable latent representations (low margin), then its model is considered less during aggregation.

Finally, federated attention vectors $\mathbf{a}^t$ are used to aggregate local weights at each t^{th} round by

$$W^{t+1} = \sum_{k \in \mathcal{K}^t} \mathbf{a}^t[k] W^t_k. \tag{11.11}$$

Algorithm 11.3 FedProto.

Input: $\mathcal{K}, T, F, W^0, \eta, N$.
for $t = 0$ **to** $T - 1$
  A server samples $\mathcal{K}^t \subseteq \mathcal{K}$ clients $\propto p_k$, and sends W^t.
  for $k \in \mathcal{K}^t$
    Compute local prototypes (11.1).
    Update W_k^t with L_k (10.2) and step size η to W_k^{t+1}.
    Compute local prototypes (11.1) and LPM (11.5).
    Send W_k^{t+1}, LPM and $\hat{\mathbf{p}}_k^t$ back to the server.
  end for
  The server computes APM (11.6), $\mathbf{a}^t$ (11.10) and W^{t+1} (11.11).
end for

Our proposed FL method which employs (11.11) to solve (10.1) is called FedProto and is summarized in Algorithm 11.3.

11.3 Theoretical Motivation of FedProto

In this section we present more in-dept theoretical insights and motivations for our approach. Our aim is to obtain a federated hypothesis function (*e.g.*, a classification or segmentation function) identified by a federated model $M(\mathcal{W}; \cdot)$ with *good generalization properties* by solving the federated optimization problem

$$\min_{W \in \mathcal{W}} L(W) = \min_{W \in \mathcal{W}} \sum_{k \in \mathcal{K}} p_k L_k(W; \mathcal{S}_k), \tag{11.12}$$

where the local objective of the k^{th} client is computed by

$$L_k(W; \mathcal{S}_k) = \frac{1}{n_k} \sum_{j=1}^{n_k} l_k(W; s_{k,j} \in \mathcal{S}_k), \tag{11.13}$$

with $l_k(\cdot; \cdot)$ being a user-specific loss function, $p_k \geq 0$ is the weight of $L_k(\cdot; \cdot)$ and $\sum_{k \in \mathcal{K}} p_k = 1$.

Margin theory has been successfully employed for characterizing hypotheses of single models, such as neural network models [423], with good generalization properties. Theoretical results show that the generalization error of classification hypotheses (classifiers) can be upper bounded by the margin of the hypotheses with respect to samples. Motivated by the success of margin theory for computing hypotheses, we consider solving (11.12) with margin maximization constraints. To this end, we first define hypothesis functions of models.

11.3.1 Hypothesis Margins

Definition 11.3.1 (Hypothesis functions)**.** A function identified by a neural network model $M(\mathcal{W}; \cdot) = C(\mathcal{W}) \circ E(\mathcal{W}; \cdot)$ parameterized by a set of weights $\mathcal{W}$ is called a model hypothesis, where $C(\mathcal{W})$ identifies a classification hypothesis (classifier) and $E(\mathcal{W})$ identifies an embedding hypothesis.

Next we define the hypothesis margin for models.

Definition 11.3.2 (Hypothesis margin)**.** The margin of the sample $s = (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ with respect to the hypothesis $M(W \in \mathcal{W}; \mathbf{x})$ for class $c \in \mathcal{C}$ is defined by

$$\theta(W, \mathbf{x}, \mathbf{y}^c) = < M(W; \mathbf{x}), \mathbf{y}^c > - \max_{b \neq c} < M(W; \mathbf{x}), \mathbf{y}^b >, \tag{11.14}$$

where $< \cdot, \cdot >$ denotes vector inner product, and $\mathbf{y}^c$ is the category label vector denoting membership of the sample to the c^{th} category. The vector $\mathbf{y}$ can be defined by a one-hot vector, where the c^{th} element of $\mathbf{y}^c$ is 1 if the sample belongs to the c^{th} class, and its other elements are equal to 0. More general codewords are proposed in [424].

Although we defined the hypothesis margin (11.14) for models $M(W; \cdot)$, the definition can be applied for defining margins of classification and embedding hypotheses.

The hypothesis margin is used to reformulate the problem (11.12) by

$$\max_{\bar{\theta}(W)} \min_{W \in \mathcal{W}} L_\Phi(W), \tag{11.15}$$

where $\bar{\theta}(W) = \min_{s=(\mathbf{x},\mathbf{y}) \in \mathcal{S}} \theta(W, \mathbf{x}, \mathbf{y})$ is the minimal margin over all the training set $\mathcal{S} = \cup_{k \in \mathcal{K}} \mathcal{S}_k$ of the model, and the loss

$$L_\Phi(W) = \Phi(\theta(W, \mathbf{x}, \mathbf{y})) \tag{11.16}$$

is defined by a function $\Phi(\cdot)$ of the margin, such as

$$\Phi(\theta(W, \mathbf{x}, \mathbf{y})) = \sum_{c \in \mathcal{C}} e^{-[< M(W;\mathbf{x}), \mathbf{y} > - < M(W;\mathbf{x}), \mathbf{y}^c >]} \tag{11.17}$$

avoiding nonlinearity of the max operator in (11.14). The following lemma shows that $\Phi(\theta(W, \mathbf{x}, \mathbf{y}))$ is lower bounded by the function of the margin $e^{-\theta(W,\mathbf{x},\mathbf{y})}$.

Lemma 1 (Lower bounding loss by hypothesis margin)**.** The loss function $L_\Phi(W)$ (11.16) is lower bounded by the function of the margin $e^{-\theta(W,\mathbf{x},\mathbf{y})}$, such that

$$L_\Phi(W) > e^{-\theta(W,\mathbf{x},\mathbf{y})}. \tag{11.18}$$

Proof. The proof follows the results given in [425]

$$\begin{aligned} L_\Phi(W) &= \sum_{c \in \mathcal{C}} e^{-[< M(W;\mathbf{x}), \mathbf{y} > - < M(W;\mathbf{x}), \mathbf{y}^c >]} \\ &= 1 + e^{-\theta(W,\mathbf{x},\mathbf{y})} \\ &\quad \sum_{\mathbf{y}^c \neq \mathbf{y}} e^{-[< M(W;\mathbf{x}), \mathbf{y} > - < M(W;\mathbf{x}), \mathbf{y}^c >] + \theta(W,\mathbf{x},\mathbf{y})} \\ &= 1 + e^{-\theta(W,\mathbf{x},\mathbf{y})} \\ &\quad \Big(1 + \sum_{\mathbf{y}^c \neq \mathbf{y}, \mathbf{y}^{c'}} e^{-[< M(W;\mathbf{x}), \mathbf{y}^{c'} > - < M(W;\mathbf{x}), \mathbf{y}^c >]}\Big) \\ &\geq 1 + e^{-\theta(W,\mathbf{x},\mathbf{y})}, \end{aligned} \tag{11.19}$$

where $c' = \arg\max_{\mathbf{y} \neq \mathbf{y}^c} < M(W; \mathbf{x}), \mathbf{y}^c >$. ■

11.3.2 Optimizing Hypothesis Margins in FL

In FL, FedAvg computes the parameters of model hypotheses at each round $t+1$ by

$$W^{t+1} = \sum_{k \in \mathcal{K}^t} \lambda_{k,t} W_k^t, \tag{11.20}$$

$$= W^t - \sum_{k \in \mathcal{K}^t} \lambda_{k,t} \psi_{k,t} \tag{11.21}$$

where W^t represents the global model parameters federately optimized at a server at round t and distributed to clients, W_k^t represents the local model parameters of client k computed at round t and sent to the server, $\psi_{k,t} \stackrel{def}{=} (W^t - W_k^t)$, $\lambda_{k,t} = \frac{n_k}{\sum_{j \in \mathcal{K}^t} n_j}$, $\sum_{k \in \mathcal{K}^t} \lambda_{k,t} = 1$, n_k is the number of samples of the client k with model parameter W_k^t, and $\mathcal{K}^t$ is the set of clients available at round t. That is, updating parameters at the server using FedAvg is equivalent to applying SGD to the "pseudo-gradient" $-\psi_t$ with learning rate 1.

At each round $t+1$, a federated optimizer optimizing hypothesis margins aims to move W by a step along the direction of the largest decrease of the loss $L(W)$ by

$$W^{t+1} = \sum_{k \in \mathcal{K}^t} \Lambda_{k,t} W_k^t, \tag{11.22}$$

$$= W^t - \sum_{k \in \mathcal{K}^t} \Lambda_{k,t} \psi_{k,t} \tag{11.23}$$

where $\sum_{k \in \mathcal{K}^t} \Lambda_{k,t} = 1$, and

$$\Lambda_{k,t} = \underset{\hat{\Lambda}_{k,t}}{\operatorname{argmin}} L_k(W_k^t, \hat{\Lambda}_{k,t}; \mathcal{S}_k) \tag{11.24}$$

$$= \underset{\hat{\Lambda}_{k,t}}{\operatorname{argmin}} \frac{1}{n_k} \sum_{j=1}^{n_k} l_k(\Delta_k^{t+1}; s_{k,j}), \tag{11.25}$$

where $\Delta_k^{t+1} = W^t - \hat{\Lambda}_{k,t} \nabla_{W_k^t} l_k(W_k^t; s_{k,j} \in \mathcal{S}_k)$.

Lemma 1 suggests that training models by minimizing (11.16) provides hypotheses with large margin. Therefore, the global parameters W^t obtained from server are moved along $\nabla_{W_k^t} l_k(W_k^t; s_{k,j})$ at the round $t+1$ to compute local parameters W_k^t by minimizing the upper bound of the margin. Thereby, we have

$$-\nabla_{W_k^t} l_k(W_k^t; s_{k,j}) = \sum_{\mathbf{x}_j \in \mathcal{X}} < M_k^{t+1}(W^{t+1}, \mathbf{x}_j), \hat{\Lambda}_{k,t}(\mathbf{x}_j) >, \tag{11.26}$$

where

$$\hat{\Lambda}_{k,t}(\mathbf{x}_j) = \sum_{c \in \mathcal{C}} \xi_c e^{\theta_k(W_k^t, \mathbf{x}_j, \mathbf{y}_j^c)}, \tag{11.27}$$

with $\xi_c = \mathbf{y}_n - \mathbf{y}_n^c$.

11.3.3 Prototype Margins

In order to make use of hypothesis margins while training models in FL, we need to first compute margins $\theta_k(W_k^t \in \mathcal{W}_k^t, \mathbf{x} \in \mathcal{X}_k, \mathbf{y} \in \mathcal{Y}_k)$ at each k^{th} client with model $M_k^{t+1}(\mathcal{W}_k^t; \mathcal{X}_k)$. Then, the

margins with target labels $\mathcal{Y}_k$ should be sent to the server for aggregation, $\forall k \in \mathcal{K}_t$. However, this approach may not be feasible due to communication complexity, and may violate privacy requirements.

To address this problem, we consider another margin of a hypothesis identified by a prototype $\mathbf{p}[c]$ as defined next.

Lemma 2 (Prototype margin of a sample (Lemma 1 in [426]))**.** Let $\mathcal{P} = \{\mathbf{p}[c] : c \in \mathcal{C}\}$ be a set of prototypes and $\mathbf{x}$ a sample point. Then the hypothesis margin of $\mathcal{P}$ with respect to $\mathbf{x}$ is $d^+ - d^-$ where

- $d^+ = \|\mathbf{p}[c] - \mathbf{x}\|_2$,
- $d^- = \|\mathbf{p}[c'] - \mathbf{x}\|_2$,
- $\mathbf{p}[c]$ is the closest prototype to $\mathbf{x}$ with the same label, and
- $\mathbf{p}[c']$ is the closest prototype to $\mathbf{x}$ with the alternative label.

Definition 11.3.3 (Normalized prototype margin of a sample)**.** The normalized prototype margin of $\mathbf{x}$ is defined by

$$\mu(\mathbf{p}[c], \mathbf{p}[c']; \mathbf{x}) = \frac{d^+ - d^-}{d^+ + d^-}. \tag{11.28}$$

The next theorem shows the relationship between the hypothesis margin θ and the prototype margin μ.

Theorem 3 (Relationship between margins for Lipschitz classifiers.)**.** Suppose that the classifier $C(\mathcal{W}; \cdot)$ is a Lipschitz map[1], $|\mathcal{C}| = O(\epsilon^{-2}\log(N))$ is the number of classes, $\epsilon \in (0, 1)$, $D \geq |\mathcal{C}|$ is the dimension of features used to compute prototypes $\mathbf{p}[c], \forall c \in \mathcal{C}$ and N is the size of the training set $\mathcal{S}$. Then, we have

$$(1 - \epsilon)\|\mu_{c,c'}\|_2 \leq \|\theta(W, \mathbf{x}, \mathbf{y})\|_2 \leq (1 + \epsilon)\|\mu_{c,c'}\|_2, \tag{11.29}$$

where $\mu_{c,c'} \stackrel{def}{=} \mu(\mathbf{p}[c], \mathbf{p}[c']; \mathbf{x})$.

Proof. We first expand $\theta(W, \mathbf{x}, \mathbf{y})$ with prototype margin by

$$\begin{aligned} \theta(W, \mathbf{x}, \mathbf{y}) &= < M(W; \mathbf{x}), \mathbf{y}^c > - < M(W; \mathbf{x}), \mathbf{y}^{c'} > \\ &= C(W; \mathbf{p}[c]) - C(W; \mathbf{p}[c']) \\ &= W_C d(\mathbf{p}[c], \mathbf{p}[c']) \end{aligned} \tag{11.30}$$

where $d(\mathbf{p}[c], \mathbf{p}[c'])$ is the Euclidean between prototypes, and $c' = \arg\max_{b \neq c} < M(W; \mathbf{x}), \mathbf{y}^b >$. We compute the norms by

$$\begin{aligned} \|\theta(W, \mathbf{x}, \mathbf{y})\|_2 &= \|W_C d(\mathbf{p}[c], \mathbf{p}[c'])\|_2 \\ &\geq \|W_C\|_2 \|d(\mathbf{p}[c], \mathbf{p}[c'])\|_2 \qquad (11.31) \\ &\geq \|W_C\|_2 \|\mu_{c,c'}\|_2, \qquad (11.32) \end{aligned}$$

where (11.31) is obtained by the Cauchy–Bunyakovsky-Schwarz inequality and (11.32) is obtained by the triangle inequality. The rest of the proof follows the Johnson-Lindenstrauss Lemma [427]. ■

[1]This assumption can be satisfied by applying spectral normalization on weights W_C of the classifier.

In the next corollary, we explore the property

$$\mathcal{P} \stackrel{def}{=} (1-\epsilon)\|\mu_{c,c'}\|_2 \leq \|\theta(W,\mathbf{x},\mathbf{y})\|_2 \leq (1+\epsilon)\|\mu_{c,c'}\|_2 \tag{11.33}$$

for Gaussian parameters (weights) of classifiers, which are usually satisfied by classifiers of deep neural networks.

Corollary 3.1 (Relationship between margins for Gaussian classifiers.)**.** Suppose that elements of the parameter of the classifier W_C are are i.i.d. from a standard normal distribution $\mathcal{N}(0, 1/|\mathcal{C}|)$. Then, the probability of preservation of norms is lower bounded by

$$Pr\Big(\mathcal{P}\Big) \geq 1 - \Omega, \tag{11.34}$$

where $\Omega = 2e^{|\mathcal{C}|(\epsilon^3-\epsilon^2)/4}$.

Proof. We first prove the upper bound for the complement of $\mathcal{P}$ by

$$Pr(\hat{\mathcal{P}}) \leq \Omega. \tag{11.35}$$

Since parameters (weights) are i.i.d. from $\mathcal{N}(0, 1/|\mathcal{C}|)$, we have

$$\begin{aligned} Pr\Big(\frac{\|\theta\|_2}{\|\mu_{c,c'}\|_2} > (1+\epsilon)\Big) &= Pr\Big(\chi^2_{|\mathcal{C}|} > (1+\epsilon)\Big) \\ &< \frac{(1-2\varrho)^{-|\mathcal{C}|/2}}{e^{\varrho|\mathcal{C}|(1+\epsilon)}} \end{aligned} \tag{11.36}$$

where $\chi^2_{|\mathcal{C}|}$ is the chi-squared distribution with $|\mathcal{C}|$ degrees of freedom with $\varrho \geq 0$, and the optimal ϱ minimizing (11.36) is $\hat{\varrho} = \frac{\epsilon}{2(1+\epsilon)}$. Thereby, we have

$$Pr(\mathcal{P}_1) < e^{|\mathcal{C}|(\epsilon^3-\epsilon^2)/4}, \tag{11.37}$$

where

$$\mathcal{P}_1 \stackrel{def}{=} \|\theta\|_2 > (1+\epsilon)\|\mu_{c,c'}\|_2. \tag{11.38}$$

We have the similar upper bound

$$Pr(\mathcal{P}_2) < e^{|\mathcal{C}|(\epsilon^3-\epsilon^2)/4} \tag{11.39}$$

for

$$\mathcal{P}_2 \stackrel{def}{=} \|\theta\|_2 < (1-\epsilon)\|\mu_{c,c'}\|_2. \tag{11.40}$$

Since $\hat{\mathcal{P}} = \mathcal{P}_1 \cup \mathcal{P}_2$), we have

$$Pr(\hat{\mathcal{P}}) < 2e^{|\mathcal{C}|(\epsilon^3-\epsilon^2)/4} \tag{11.41}$$

and

$$Pr(\mathcal{P}) \geq 1 - 2e^{|\mathcal{C}|(\epsilon^3-\epsilon^2)/4} \tag{11.42}$$

$$\geq 1 - \Omega. \tag{11.43}$$

■

Corollary 3.1 provides a lower bound on the probability of preserving margins of hypotheses

and prototypes assuming that the distribution of weights of classifiers is a scaled Gaussian $\mathcal{N}(0, 1/|\mathcal{C}|)$. However, this result is also valid for weights i.i.d. from unscaled Gaussian $\mathcal{N}(0, 1)$ as proposed in the following corollary.

Corollary 3.2 (Relationship between margins for unscaled Gaussian classifiers.)**.** Suppose that elements of the weights of the classifier W_C are i.i.d. from a standard normal distribution $\mathcal{N}(0, 1)$. Then, we have

$$Pr\Big(\mathcal{P}_u\Big) \geq 1 - \Omega, \tag{11.44}$$

where

$$\mathcal{P}_u \stackrel{def}{=} \frac{C}{D}\|\mu_{c,c'}\|_2 \leq \|\theta(W, \mathbf{x}, \mathbf{y})\|_2 \leq \frac{D}{C}\|\mu_{c,c'}\|_2. \tag{11.45}$$

Proof. The first part of the proof applies Theorem 3 and Corollary 3.1 for proving the property $\mathcal{P}_u$ for unscaled Gaussian weights. The second part of the proof applies Corollary 3.1 for proving the inequality (11.44). ■

Theorem 3, Corollary 3.1 and Corollary 3.2 show that the norm of prototype margin is close to the norm of hypothesis margin by $\epsilon \in (0, 1)$. In order words, distance between hypotheses diverge from the distance between prototypes by $1 \pm \epsilon$ with probability lower bounded by a function of ϵ. We employ this result to lower bound the error $L_\Phi(W)$ in the following theorem.

Theorem 4 (Lower bounding loss by norm of prototype margin)**.** The loss function $L_\Phi(W)$ (11.16) is lower bounded by a function of the prototype margin $e^{-\|\mu_{c,c'}\|_2^2}$ for any two prototypes with $c \neq c'$, such that

$$L_\Phi(W) > e^{-\|\mu_{c,c'}\|_2^2}. \tag{11.46}$$

Proof.

$$\begin{aligned} L_\Phi(W) &= \sum_{c \in \mathcal{C}} e^{-[<M(W;\mathbf{x}),\mathbf{y}> - <M(W;\mathbf{x}),\mathbf{y}^c>]} \\ &= 1 + e^{-\|\theta\|_2^2} \sum_{\mathbf{y}^c \neq \mathbf{y}, \mathbf{y}^{\hat{c}}} e^{-[A-B-G^2]} \\ &\geq e^{-\|\theta\|_2^2}, \end{aligned} \tag{11.47}$$

where $\theta \stackrel{def}{=} \theta(W, \mathbf{x}, \mathbf{y})$,

$$A \stackrel{def}{=} < M(W;\mathbf{x}), \mathbf{y} > \tag{11.48}$$

$$B \stackrel{def}{=} < M(W;\mathbf{x}), \mathbf{y}^c > \tag{11.49}$$

$$G \stackrel{def}{=} < M(W;\mathbf{x}), \mathbf{y}^{\hat{c}} > \tag{11.50}$$

with $\hat{c} = \arg\max_{b \neq c} < M(W;\mathbf{x}), \mathbf{y}^b >$. By Theorem 3, Corollary 3.1 and Corollary 3.2, we have

$$e^{-\|\theta\|_2^2} \geq e^{-\|\mu_{c,c'}\|_2^2} \tag{11.51}$$

which implies

$$L_\Phi(W) \geq e^{-\|\mu_{c,c'}\|_2^2}. \tag{11.52}$$

■

In [425] Saberian and Vasconcelos study the relationship between hypothesis margin, and different types of loss functions such as logistic and exponential loss. These results are used to lower bound different loss functions by hypothesis margins, which can be employed to bound these loss functions by prototype margins. Since this analysis is beyond the scope of this work, we consider applying Theorem 4 for different loss functions as a future work.

11.3.4 Federated Learning with Prototype Margins

Similar to hypothesis margins, Theorem 4 proposes that minimizing the loss $L_\Phi(W)$ provides models with large prototype margin. While training models, we employ this result to minimize training loss by maximizing prototype margins by

$$W^{t+1} = \sum_{k \in \mathcal{K}^t} \mathbf{a}^t[k] W_k^t \tag{11.53}$$

$$= W^t - \sum_{k \in \mathcal{K}^t} \mathbf{a}^t[k] \psi_{k,t} \tag{11.54}$$

where $\sum_{k \in \mathcal{K}^t} \mathbf{a}^t[k] = 1$. The weight attention functions of margins $\mathbf{a}^t[k], \forall k, t$, are estimated by

$$\mathbf{a}^t[k] = \underset{\hat{\mathbf{a}}^t[k]}{\operatorname{argmin}} L_k(W_k^t, \hat{\mathbf{a}}^t[k]; \mathcal{S}_k) \tag{11.55}$$

$$= \underset{\hat{\mathbf{a}}^t[k]}{\operatorname{argmin}} \frac{1}{n_k} \sum_{j=1}^{n_k} l_k(\Delta_k^{t+1}; s_{k,j}), \tag{11.56}$$

where $\Delta_k^{t+1} = W^t - \hat{\mathbf{a}}^t[k] \nabla_{W_k^t} l_k(W_k^t; s_{k,j} \in \mathcal{S}_k)$.

The maximum hypothesis margins $\Lambda_{k,t}, \forall t$ can be computed by minimizing loss $L_\Phi(W)$ at each client as proposed in Lemma 1. Similarly, Theorem 4 shows that optimizing parameters of local models by minimizing the loss $L_\Phi(W)$ enables models to learn features with large prototype margins at each round t. Then, local hypothesis margins $\Lambda_{k,t}, \forall k, t$ can be linearly aggregated in distributed linear models such as distributed boosting, as discussed in Section 11.3.2. However, linear aggregation of margins may provide biased estimations for nonlinear or biased models, such as models of deep neural networks trained with non-i.i.d. data, in FL. Therefore, we propose two novel margin definitions to compute $\hat{\mathbf{a}}^t[k], \forall k, t$:

- In order to incorporate information on the evolution of prototypes among different rounds $t-1$ and t, we define Local Prototype Margins (LPMs). The LPMs $\mu_{loc,k}^t, \forall k, t$, measure change of local prototypes obtained from local models before and after local training.
- In order to incorporate information on discrepancy between local and global (aggregate) set of prototypes, we define Aggregate Prototype Margins (APMs).
  The LPMs and APMs are then combined to approximate $\hat{\mathbf{a}}^t[k], \forall k$ at each round t.

11.4 Experimental Setup

Federated Datasets. We evaluate on various tasks, models and real-world federated vision datasets. For all the considered datasets, we randomly split the data on each local client into a training and a testing set with a 80/20 ratio. Full details on the experimental setup have been introduced in Section 10.3.

Implementation Details. Utilized hyper-parameters are reported on the right side of Table 10.1. We tuned learning parameters of each dataset on FedAvg (with $F = 1$ and no system

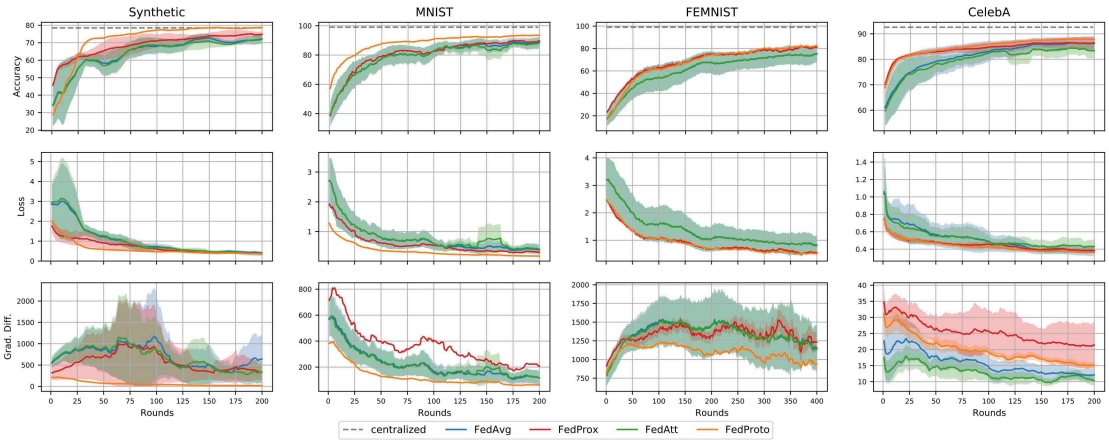


Figure 11.2: Experimental results for the classification task. Evaluation is performed across $\delta \in \{0\%, 50\%, 80\%\}$ and a moving average window of 10% rounds is applied for visualization. Solid lines and shaded regions represent the mean and standard deviation, respectively.

heterogeneity) and, for fair comparison, we use the same parameters on all experiments for that dataset. We set $|\mathcal{K}^t| = 10, \forall t$ for all datasets. Randomly selected clients and mini-batch orders are kept fixed across all runs for comparative experiments. For simplicity, we use a constant learning rate on classification tasks, and polynomially decaying learning rate with power 0.9 and weight decay $4 \cdot 10^{-5}$ [4,5] for segmentation tasks. For classification data, we measure accuracy as the percentage of correct predictions, whilst for segmentation data we use the mean Intersection over the Union (mIoU). All simulations are performed for 10% rounds more, and metrics are moving averaged over a window of 10% rounds in the visualization.

We fix the number of selected clients to be 10 for all experiments and most of the hyper-parameters has been reported in Table 10.1. Unless otherwise stated, we assume that FedAvg does not tolerate partial local solutions (*i.e.*, dropped clients are not aggregated), while FedProto and FedProx do tolerate them. For Synthetic, MNIST, FEMNIST, and CelebA, we set the proximal loss term of FedProx following the guidelines of [375], and the best accuracy is obtained respectively for: 0.1, 1, 1, 0.01.

We developed our framework in Tensorflow [125]. We simulate the federated learning setup (1 server and $|\mathcal{K}|$ clients) on a single NVIDIA GeForce RTX 2080 Ti GPU with 2 Intel Xeon Gold 5220 CPU at 2.20GHz.

11.5 Experimental Analyses for Federated Vision

11.5.1 Federated Image Classification

In this section, we report an extensive evaluation of our approach on image classification tasks. We compare FedProto with the baseline FedAvg, with the state-of-the-art regularization-based FedProx [375] and with FedAtt [379], which employs weight-based attention. Figure 11.2 shows per-round aggregate accuracy, training loss and gradients difference on the four classification datasets introduced in Section 11.4. From the first row of Figure 11.2, we observe that FedProto robustly outperforms FedAvg in terms of accuracy on every dataset. FedAtt brings minimal improvement compared to FedAvg, proving that a simple weight-based attentive mechanism is not very useful in vision tasks. FedProx, instead, leverages accuracy thanks to the toleration of

partial workload and presence of the proximal term. However, our approach can effectively match or surpass the accuracy of FedProx. Additionally, we observe that both FedProto and FedProx show much lower variance (narrower shaded region) than competing approaches by tolerating partial results. Similar considerations are also reflected on the training loss (second row). The third row reports the average of squared ℓ_2 norm of difference of gradients over all clients, *i.e.*, $\frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}||\nabla L_k(W^t;\mathcal{S}_k) - \nabla L(W^t)||_2^2$. As in [375], we interpret this dissimilarity measure as a proxy of accuracy. In particular, we observe how FedProto shows smaller dissimilarity (*i.e.*, better convergence [375]) compared to FedProx, thanks to the regularization effects brought by the proposed modules. To better appreciate accuracy and loss gaps observed in Figure 11.2, we give results obtained using the aggregate models at the final round in Table 11.1 where FedProto shows significant improvements across all the datasets.

Ablation Studies. To explore the effect of the components of our approach on accuracy, we report a comparative ablation study in Table 11.2. First, we noticed that our approach tends to produce weights $\mathbf{a}^t$ deviating less from a fairness policy (*i.e.*, aggregation of weights W_k^t by $\mathbf{f}^t[k] = 1/K', \forall k, \forall t$) than FedAvg, as also observed in other contemporary approaches [381,391]. A fair policy (row 2), indeed, outperforms FedAvg by a small margin. However, we argue that this is an implicit effect of the weighted sampling scheme of the active clients at each round introduced in Section 10.2. As a matter of fact, sampling active clients i.i.d. (row 3) brings results comparable to FedAvg. Second, we analyse the toleration of partial workload. Adding it on top of naïve implementations of FedAvg and Fairness (rows 4 and 5) improves the accuracy and the robustness over different amounts of δ. At the same time, we remark that our approach can achieve competitive performance even without tolerating partial workload (row 6).

Analyses of our model design are given in the last block of Table 11.2. Margin-based deviation can be viewed as an enhanced measure rather than just using the distance between prototypes belonging to the same class, *i.e.*, using only d^+ (row 7) from (11.2) to accommodate the class-wise probability distribution obtained from the distributed clients during aggregation. Although providing considerable improvements compared to FedAvg, we found margin-based deviation to be generally more stable. Finally, employing only one of the two proposed clues (LPM and APM in rows 8 and 9) still improves accuracy, and the combination of the two (last row) outperforms the effect of the singular components.

Table 11.1: Final mean and std of accuracy (%) and loss from Figure 11.2. Centralized accuracy are $78.5, 99.0, 99.4, 92.6$, and losses are $0.33, 0.00, 0.00, 0.15$ for Synth., MNIST, FEMNIST and CelebA.

		FedAvg	FedProx	FedAtt	FedProto
Accuracy	Synthetic	72.3 ± 2.6	74.8 ± 1.6	72.1 ± 2.7	$\mathbf{78.7 \pm 0.2}$
	MNIST	88.8 ± 3.8	$91.7 \pm \mathbf{0.2}$	88.4 ± 3.7	$\mathbf{93.3 \pm 0.2}$
	FEMNIST	75.1 ± 7.7	81.1 ± 1.0	75.5 ± 7.5	$\mathbf{82.5 \pm 0.3}$
	CelebA	86.2 ± 2.8	86.4 ± 2.4	83.4 ± 3.0	$\mathbf{87.8 \pm 0.4}$
Loss	Synthetic	0.41 ± 0.06	0.37 ± 0.07	$\mathbf{0.36} \pm 0.12$	$\mathbf{0.36 \pm 0.02}$
	MNIST	0.39 ± 0.17	$0.30 \pm \mathbf{0.02}$	0.41 ± 0.16	$\mathbf{0.18 \pm 0.02}$
	FEMNIST	0.83 ± 0.35	0.55 ± 0.04	0.81 ± 0.34	$\mathbf{0.51 \pm 0.01}$
	CelebA	0.38 ± 0.06	0.39 ± 0.03	0.43 ± 0.08	$\mathbf{0.36 \pm 0.02}$

Aggregate Mean Margin (AMM). To examine margin maximization properties of federated optimizers, we define a measure called aggregate mean margin (AMM) by

$$\bar{\boldsymbol{\mu}}[t] = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \mu(\mathbf{p}_{agg}^t[c], \mathbf{p}_{agg}^t). \tag{11.57}$$

Table 11.2: MNIST classification accuracy (%) of different strategies.

	δ			
Method	0%	50%	80%	Avg. $\pm$ Std.
FedAvg	92.7	88.7	85.1	88.8 ± 3.8
Fairness	92.8	89.9	86.5	89.7 ± 3.2
Fairness sampling i.i.d.	92.5	88.4	84.9	88.6 ± 3.8
FedAvg + toleration	92.7	90.2	89.1	90.7 ± 1.8
Fairness + toleration	92.8	91.2	90.6	91.5 ± 1.1
FedProto (no toleration)	**93.5**	90.8	88.1	90.8 ± 2.7
FedProto (d^+) only	92.8	92.4	92.1	92.4 ± 0.4
FedProto (APM only)	93.0	92.7	92.6	$92.8 \pm \mathbf{0.2}$
FedProto (LPM only)	91.9	91.0	90.6	91.2 ± 0.7
FedProto	**93.5**	**93.4**	**93.1**	$\mathbf{93.3 \pm 0.2}$

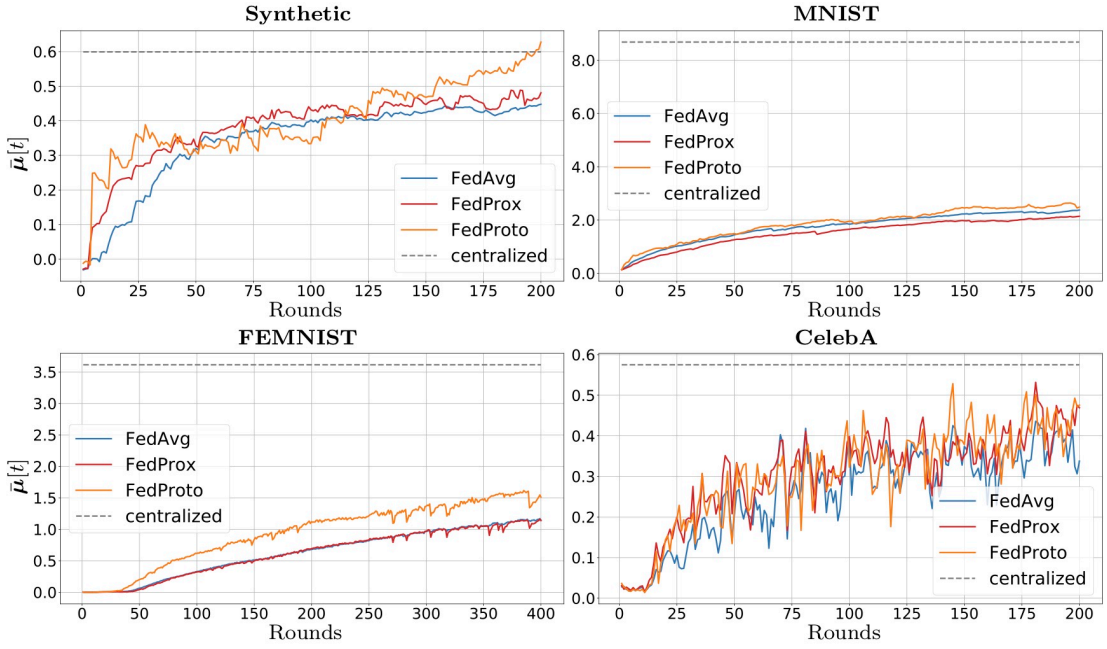


Figure 11.3: Per-round AMM ($\bar{\boldsymbol{\mu}}[t]$) values on classification datasets.

In Figure 11.3, we show change of AMM for different optimizers and datasets during training in FL. FedProto achieves higher $\bar{\boldsymbol{\mu}}[t]$ compared to other optimizers. This is a direct consequence of a better shaping of latent representations with improved class-discrimination acting as regularizer for learning meaningful feature representations similar to centralized training. The AMM for the last round, $\bar{\boldsymbol{\mu}}[T]$, is reported in Table 11.3. The results show a positive correlation between AMM and accuracy (given in Table 11.1) with Pearson's correlation coefficient $\rho = 0.68$ (p-value 0.01).

Federated Feature Discrepancy (FFD). FFD is devised to analyze how feature distributions provided by a model M_A trained with a federated optimizer A are closer to those generated by centralized training of a model M_C, compared to a baseline optimizer B. To this end, we first compute distribution $P_k^t, \forall k$ of features provided by M_A. Second, we train a model M_C on the same dataset without any distributed setting, and Q denotes the distributions obtained from M_C. Then, we compute the average Maximum Mean Discrepancy (MMD) [428] between

Table 11.3: Margin $\bar{\mu}[T]$ of the final aggregate model and FFD (%).

		Synthetic	MNIST	FEMNIST	CelebA
$\bar{\mu}[T]$	FedAvg	0.45	2.38	1.15	0.34
	FedProx	0.48	2.14	1.14	0.47
	FedProto	**0.63**	**2.49**	**1.51**	**0.48**
	Centralized	0.60	8.68	3.61	0.48
FFD	FedProx	61.9	5.9	1.5	4.5
	FedProto	**64.5**	**7.7**	**5.7**	**7.8**

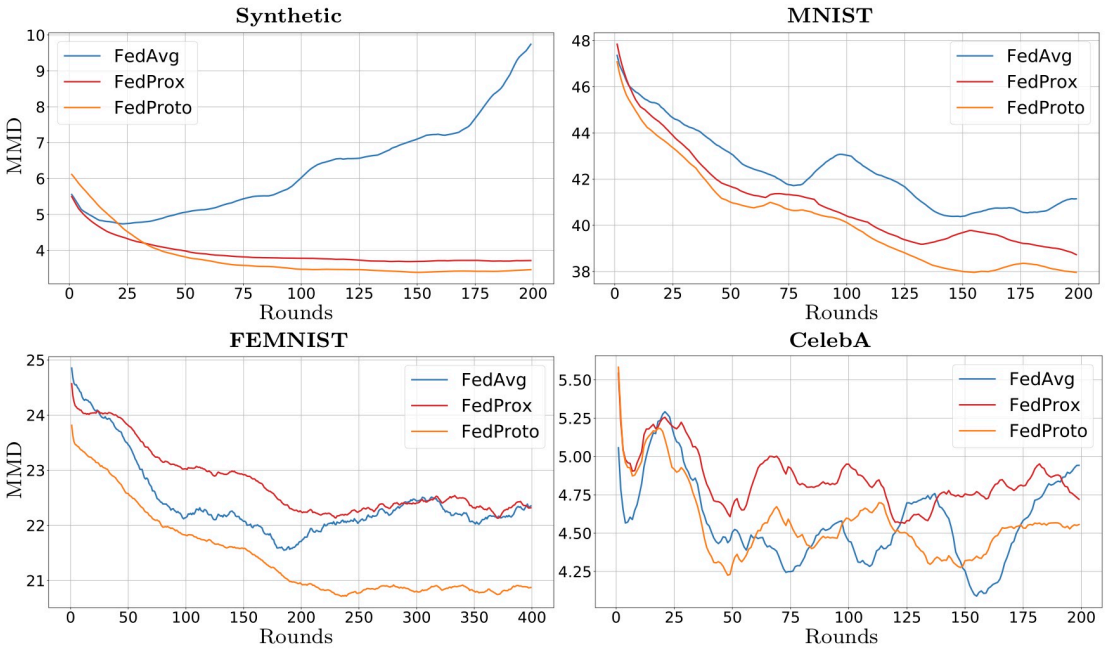


Figure 11.4: Per-round MMD from Eq. (11.58) on classification datasets. Higher MMD indicates features of the FL algorithm to be more similar to the features learned in centralized training.

P_k^t and Q by

$$MMD_A^t = \frac{1}{|\mathcal{K}^t|} \sum_{k \in \mathcal{K}^t} MMD(P_k^t, Q). \tag{11.58}$$

We define the FFD (%) between A and B as the relative gain of MMD_A^T over MMD_B^T by

$$FFD(A, B) = \frac{MMD_B^T - MMD_A^T}{MMD_B^T} \times 100. \tag{11.59}$$

Since our interest is to give a comparison with respect to the baseline FedAvg, we set A to FedProx or FedProto and B to FedAvg. The per-round MMD is shown in Figure 11.4 and the final FFD values are reported in the bottom part of Table 11.3. Overall, we observe that distributions of features learned using FedProto are consistently more similar to those learned in centralized training than FedAvg: the higher is the MMD and the closer the features produced by an FL algorithm are to those learned in centralized training. Last, we also note that FedProx can achieve some latent regularization thanks to the proximal term, however it is robustly surpassed by our proposed FedProto.

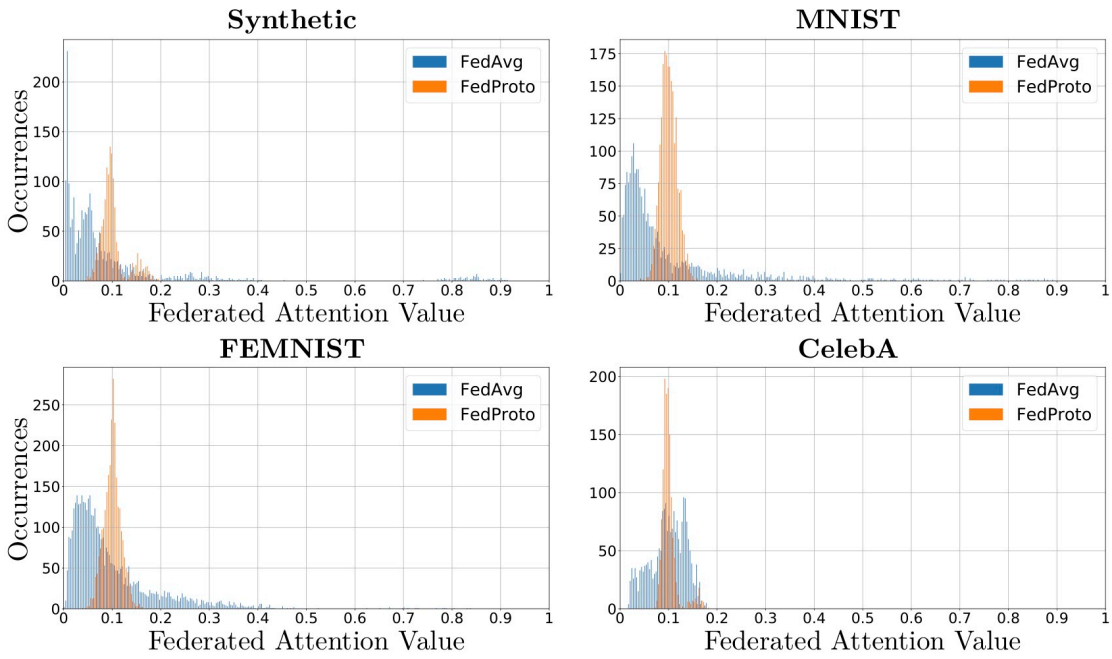


Figure 11.5: Comparison of distributions of the federated attention vector $\mathbf{a}^t[k], \forall k, \forall t$, on classification datasets for FedAvg and our FedProto. FedProto produces attention values having a much lower variance from the average value ($|\mathcal{K}| = 10$ is used) compared to FedAvg. FedAvg weights, instead, follow the distribution of the number of samples, which could lead the framework to ignore clients with less samples during aggregation, regardless of the statistical distribution of local samples.

11.5.2 Federated Attention Values in Image Classification

We mentioned that our approach tends to produce federated attention values $\mathbf{a}^t[k]$ deviating less from a fairness policy (*i.e.*, aggregation of weights W_k^t by $\mathbf{f}^t[k] = 1/K', \forall k, \forall t$) than FedAvg, as also observed in other contemporary approaches [381, 391]. In Figure 11.5, we compare the distribution of federated attention values $\mathbf{a}^t[k], \forall k, \forall t$, of FedAvg and of our approach. The results show that FedProto produces attention values having a much lower variance from the average value (we remark that $|\mathcal{K}| = 10$ is used) compared to FedAvg. In particular, FedAvg weights follow the same distribution of the number of samples, which could lead the framework to ignore clients with less samples during aggregation, regardless of the statistical distribution of local samples. Moving from these considerations, we found a fairness policy to have comparable results to FedAvg (see Section 10.4) and to be significantly surpassed by our approach.

11.5.3 Federated Semantic Segmentation

We analyze our FedProto for federated semantic segmentation. Differently from image classification, segmentation task is more challenging as it involves dense predictions and highly class-imbalanced datasets. Altogether, these circumstances make aggregating local models even more severe.

We start by analyzing the effect of i.i.d. structure (i.i.d.-ness) of data on mIoU of federated segmentation models. For this purpose, we distribute two benchmark datasets among clients using the Dirichlet distribution with concentration parameter α (details are given in Section 11.4 and in Suppl. Mat.). Then, we train models on distributed data using the baseline FedAvg and

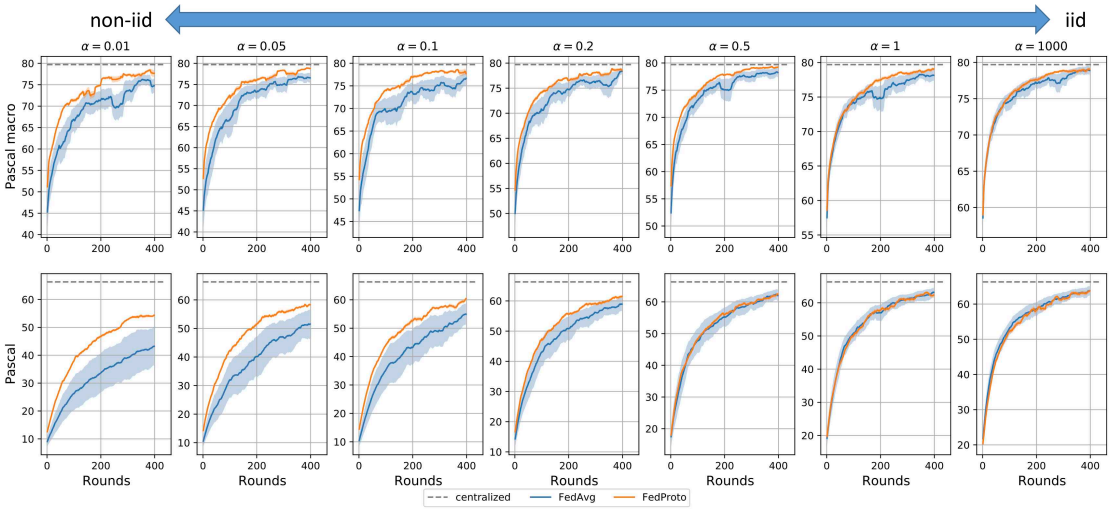


Figure 11.6: Change of mIoU on segmentation data distributed using different α values. Evaluation is performed across $\delta \in \{0\%, 50\%, 80\%\}$ and a moving average window of 10% rounds is applied. Solid and shaded lines represent mean and standard deviation.

our FedProto. The results depicted on Figure 11.6 show the relationship between convergence of models and i.i.d.-ness of data. Note that, as the non-i.i.d.-ness of distributed data increases by lower α, data heterogeneity and client drift increase. Our FedProto improves mIoU and robustness compared to FedAvg on every configuration, and especially on highly non-i.i.d. data, where class-conditional representations on certain remote clients could be non-reliable due to the non-i.i.d. partitioning (only few samples for particular classes observed on certain clients).

A qualitative analysis on segmentation and entropy maps of two sample images comparing the final aggregate models of FedAvg and FedProto on different data splitting configurations (*i.e.*, setting $\alpha \in \{0.01, 0.1, 1\}$) is reported in Figure 11.7.

Segmentation maps: Output segmentation maps (rows 1, 4 and 7) improve when data are more i.i.d., better resembling segmentation maps produced by centralized training. FedProto significantly outperforms FedAvg for more non-i.i.d. data ($\alpha = 0.01$): the cat in first row and the horse in fourth row are correctly labeled and well-defined, whilst FedAvg labels them as a mixture of other animals. The ability to distinguish between class ambiguity is the direct consequence of a better latent space organization and regularization that FedProto achieves by maximizing prototype margin.

Entropy maps: Second, we report the entropy map of the softmax probabilities of the final model (rows 2, 5 and 8): *i.e.*, $H_S = H(M(\mathcal{W}^T; \mathcal{X}))$, with $H(\cdot)$ being the pixel-wise Shannon entropy [151,429]. Low entropy (dark blue) indicates a peaked distribution which is the reflection of high confidence of the network on its prediction, and vice-versa. Ideally, the entropy should be low for every pixel. However, as we can observe from centralized training, contours of objects and certain regions of the images (*e.g.*, the mane of the horse in row 4) have high entropy due to uncertainty on the precise edge localization of the objects or due to intrinsic ambiguity with other classes (all considered animal classes have fur with similar pattern). With these considerations in mind, we observe how FedProto produces generally darker entropy maps than FedAvg, especially on non-i.i.d. data. Last, we analyze the feature-level entropy maps upsampled to match input resolution (rows 3, 6 and 9). To compute it, features $E(\mathcal{W}^T; \mathcal{X})$ are first normalized to $\hat{E}(\mathcal{W}^T; \mathcal{X})$, such that the sum over the channels at each low-resolution pixel location is 1 (*i.e.*, in order for

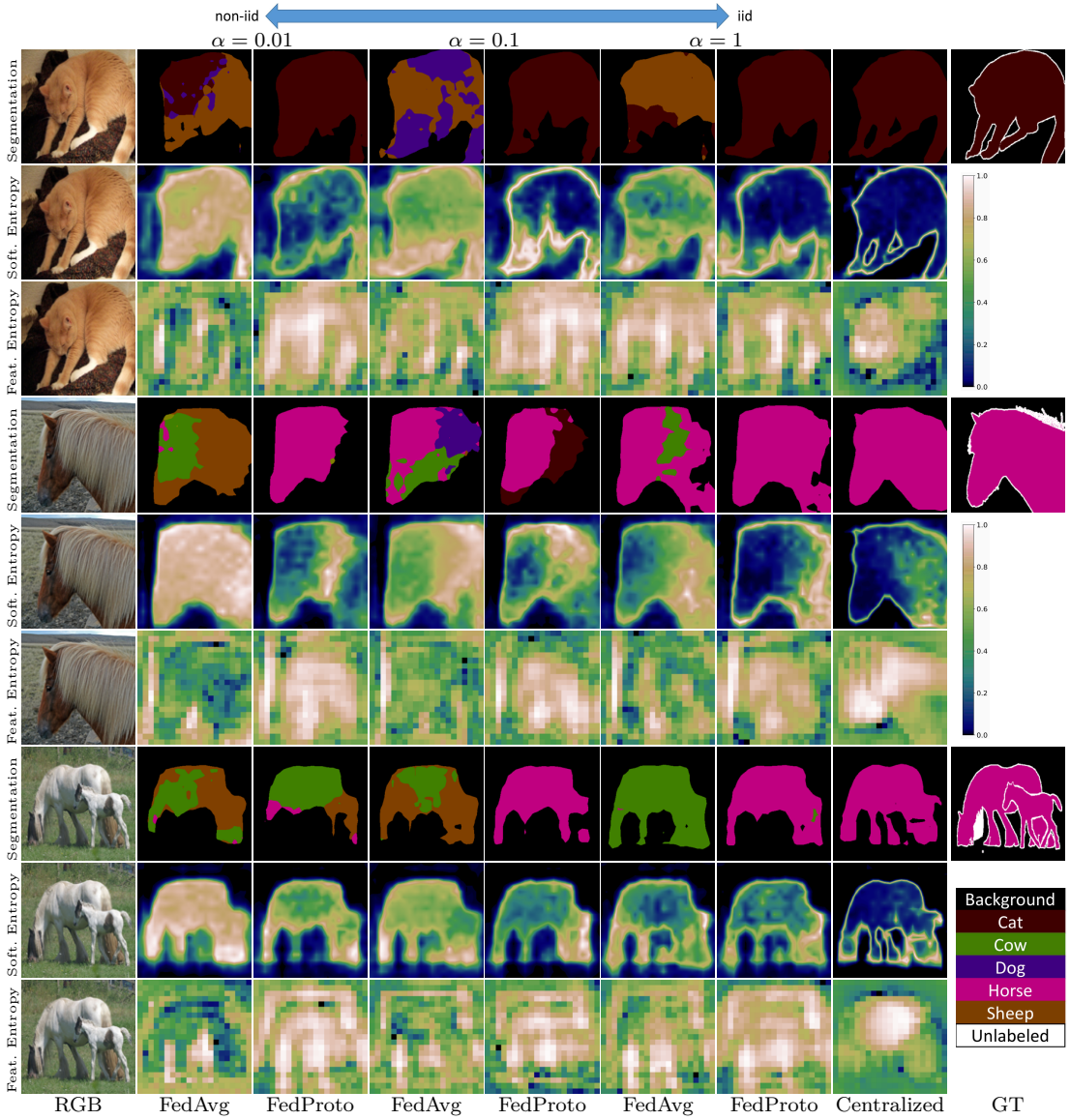


Figure 11.7: Qualitative results for models trained using FedAvg and FedProto using three non-i.i.d. to i.i.d. configurations of Pascal VOC2012 dataset. For each of the three sample images, we depict; the output segmentation map (rows 1, 4 and 7), the softmax-level entropy map (rows 2, 5 and 8), and the feature-level entropy map (rows 3, 6 and 9). As a reference, output maps of models obtained using centralized training are shown on the second last column.

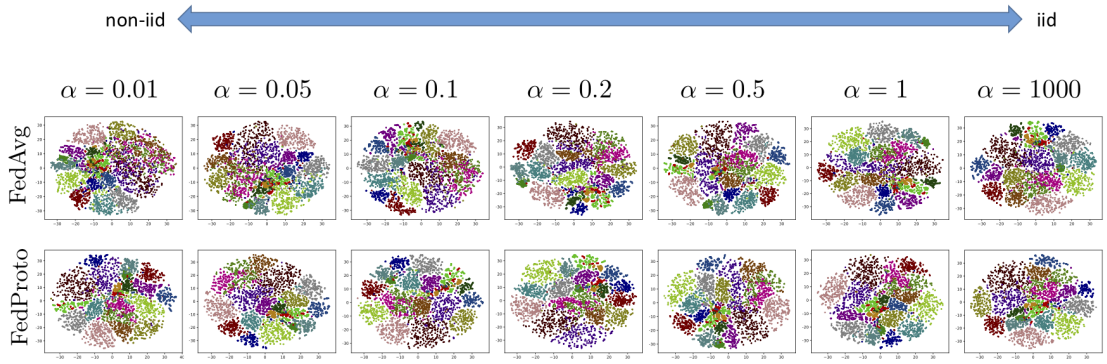


Figure 11.8: Comparison of t-SNE embedding plots of feature representations learned by FedAvg and by our FedProto, using the Pascal VOC 2012 segmentation benchmark with 20 object level classes. Analyses are performed over different values of α. The *background* class is not included in the visualization, and the colors refer to the Pascal VOC2012 colormap.

them to be considered as probability vectors), and then we define $H_E = H(\hat{E}(\mathcal{W}^T; \mathcal{X}))$. In this case, H_E measures how representative a feature is at each pixel location. Ideally, features corresponding to the desired class should be well activated so that the decoder can discriminate between them and assign the correct label: this is the case of centralized training where features corresponding to (certain parts of) the object class are bright (*i.e.*, high entropy denoting many activated patterns). We observe that FedProto produces a feature-level entropy map which is more similar to centralized training than the map produced by FedAvg (particularly visible for low α values).

One of the main effects of our proposed FedProto is a class-conditional latent-level regularization, achieved via prototype guided federated optimization and margin maximization of the aggregate model during its distributed training. In order to visually represent the main effects, we report in Figure 11.8 the 2D t-SNE embeddings of the features of the final aggregate model [367] for different values of α. Here, the background class is not included and the colors refer to the Pascal VOC2012 colormap. Class membership for the low-resolution feature map is obtained with simple nearest neighbor downsampling of the full-resolution segmentation maps. By visual inspection, we observe that t-SNE embeddings produced using the final aggregate model from FedProto are better subdivided into clusters (*i.e.*, points of the same color). In particular, for $\alpha = 0.01$, FedAvg confuses some animal classes (horse in pink, sheep in brown, cow in green, cat in dark red and dog in purple) into one mixed point cloud on the top right part of the plot, lacking class-discrimination at the feature level. FedProto, instead, produces a much clearer separation among these (and others) classes, being able to build class-discriminative clusters at the latent level (*i.e.*, clusters points on the basis of their class membership). Similar discussion can be made also for the remaining scenarios, with a progressively smaller difference between t-SNE embeddings produced by FedAvg and FedProto as the data i.i.d.-ness increases.

11.6 Conclusions and Future Work

In this section, we proposed FedProto, a distributed machine learning paradigm for vision models that can handle clients characterized by system and statistical heterogeneity. Previous approaches disregard internal representations to aggregate model weights. FedProto, instead, com-

putes client deviations based on the inner class-conditional prototypical representations and uses them to drive federated optimization using an attentive mechanism. The experimental analyses across a suite of federated datasets on both classification and semantic segmentation demonstrated the effectiveness of our framework on both classification and segmentation datasets. In particular, we established a new benchmark on federated semantic segmentation task outlining a new research direction. We hope that our problem formulation and our approach will encourage future works on this novel research topic.

12 Conclusions and Future Directions

12.1 Conclusions

This thesis has investigated the capability of deep neural network to adapt to changes in the domain distribution with a last insight on distributing model training across decentralized clients. We started our analyses from adaptation to changes in the label space (continual learning), followed by a study on progressive refinement of the label space over time (coarse-to-fine learning), then we moved to analyze adaptation to unseen domain distributions without exploiting any labels (unsupervised domain adaptation), and finally we devoted some attention to distributed model training (federated learning). We undertook a comprehensive path touching many different aspects toward distributed, ubiquitous and practical deployment of complex artificial vision systems, focusing on the semantic segmentation task.

In the first part of the thesis, we introduced the problem of continual learning of novel semantic concepts in dense labeling tasks. While traditional deep learning models assume that all the data samples are available during the training phase and that the training is performed on the entire dataset, we addressed the need of training deep neural networks on sequential tasks with samples progressively available over time. A key trade-off emerged between overcoming intransigence (*i.e.*, inability) of the model of learning new tasks and limiting forgetting of past tasks. We tackled these issues proposing various regularization-based techniques ranging from knowledge distillation, parameter freezing, latent geometrical regularization and replay techniques. First, we investigated how knowledge of previous tasks can be transferred to models of subsequent tasks at different levels of the deep neural networks, and we observe that the encoder parameters are less sensible to changes of the label-space data distribution. Second, we presented some latent representation shaping techniques (such as prototype matching, feature sparsification and contrastive-based objectives) to prevent forgetting whilst simultaneously improving the recognition of novel classes. Last, we sampled replay data either from a conditional GAN or from the web and we used them to alleviate catastrophic forgetting and background inpainting to mitigate background shift.

In the second part of the thesis, we defined and tackled the coarse-to-fine learning of new semantic concepts, where a deep learning model initially trained on a coarse set of classes is updated to recognize either more refined categories than those originally introduced (semantic level coarse-to-fine), or sub-parts belonging to the object-level class (spatial level coarse-to-fine). In the semantic level coarse-to-fine scenario, we have explored multiple ways of feeding the output

of the coarse-level model to the fine-level model increasing the overall accuracy on the finer set of classes. In the spatial level coarse-to-fine scenario, we refined the stacking of multiple model architectures by feeding an embedded version of the output of the object-level (*i.e.*, coarse-level) model at the decoding stage of the fine-level model. This refinement guarantees higher object-level semantic conditioning and guidance during the fine-level decoding stage. This architecture was further reinforced by an auxiliary reconstruction module which reconstructs object-level classes from part-level predictions, penalizing when part-level predictions are located outside the respective object-level class. Finally, a graph matching constraint maintain the reciprocal relationships between parts.

The third part studied the unsupervised domain adaptation task in semantic segmentation, where a deep neural network trained on a source domain distribution is adapted to a target domain distribution using only target samples without labeling information. To address this task, we first categorized all the approaches proposed in the literature outlining three main levels to which adaptation may occur: namely, at the output level, at the input level and at the feature level. We investigated a few methods covering all the adaptation levels. First, we proposed an output-level adversarial learning framework based on fully-convolutional discriminators generating a confidence map of the segmentation output, which is later used for self-teaching. A region growing module refined the confidence maps on the basis of the segmentation output on real-world images. Then, our framework was enriched by class-aware and time-varying confidence thresholds to adapt to different classes and different stages of learning of the semantic segmentation network. Second, we investigated input-level alignment by means of cyclic domain transfer. Finally, feature-level domain alignment was enforced by using either latent-level discriminators or latent-space shaping objectives enforcing features clustering, orthogonality, sparsity and norm alignment. Additionally, we also showed how the proposed techniques are complementary to other strategies in the literature.

The fourth and final part of the thesis, instead, was related to federated learning of deep learning models. In this framework, data is available only at local clients and cannot be shared with a central server; hence, model training is performed decentralized at each client and then the solutions are shared with a central server and aggregated. First, we proposed a simple federated aggregation scheme which is fair from the the users perspective and we showed how it can improve with respect to the baseline FedAvg both the final accuracy value and the convergence rate, whilst reducing at the same time fluctuations of accuracy of the aggregate model. Second, we developed a more advanced aggregation scheme on the basis of client deviations computed from inner class-conditional prototypical representation and we used them via an attentive mechanism. Thanks to the guidance provided by the learned internal representations we were the first to evaluate on semantic segmentation benchmarks, paving the way to decentralized deep scene understanding.

12.2 Open Problems and Future Directions

While automatic semantic scene understanding has been a particularly active field in the last few years, there are several research directions still relatively unexplored. In this thesis we walked through some of the current trending research directions in the computer vision community related to semantic scene understanding. Most of the proposed techniques is completely (or partially) agnostic with respect to the deep network architecture, to the task and to the dataset employed for evaluation. Hence, future research will focus on evaluating even more the general applicability of the algorithms in different contexts.

Furthermore, the accuracy on some single tasks (*e.g.*, UDA alone, or CL alone) is reaching satisfactory results and novel requirements (or tasks) are starting to emerge. Often, new tasks

are an evolution of previous ones and so techniques could be reused and extended to address the novel requirements. For instance, hybrid approaches combining UDA and CL can overcome the need for multiple changes in domains and tasks over time. Adding FL to the process we would be able to cope with the real-world variability locally, without the need for sharing large amount of data to a central server and reducing privacy concerns. These and other tasks are of paramount importance toward the effective deployment of versatile visual systems in the wild.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[2] F. Yu, V. Koltun, and T. A. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 636–644.

[3] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[4] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 833–851.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, pp. 834–848, 2018.

[6] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[7] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[8] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[9] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 1089–1106, 2019.

[10] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[11] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information fusion*, vol. 58, pp. 52–68, 2020.

[12] X. Ren, L. Xie, C. Wei, S. Qiao, C. Su, J. Liu, Q. Tian, E. K. Fishman, and A. L. Yuille, "Generalized coarse-to-fine visual recognition with progressive training," *arXiv preprint arXiv:1811.12047*, 2018.

[13] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: A review," *Technologies*, vol. 8, no. 2, 2020.

[14] M. Mancini, "Towards recognizing new semantic concepts in new visual domains," *arXiv preprint arXiv:2012.09058*, 2020.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017, pp. 1273–1282.

[16] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Advances in Neural Information Processing Systems Workshops (NeurIPSW)*, 2015.

[18] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[19] U. Michieli, M. Toldo, and P. Zanuttigh, "Unsupervised Domain Adaptation and Continual Learning in Semantic Segmentation," *Advanced Methods and Deep Learning in Computer Vision, Elsevier*, 2021.

[20] U. Michieli and P. Zanuttigh, "Knowledge distillation for incremental learning in semantic segmentation," *Computer Vision and Image Understanding*, vol. 205, p. 103167, 2021.

[21] ——, "Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[22] A. Maracani*, U. Michieli*, M. Toldo*, and P. Zanuttigh, "Generative replay for continual learning in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[23] F. Cermelli, M. Mancini, S. R. Bulò, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] M. Mel, U. Michieli, and P. Zanuttigh, "Incremental and multi-task learning strategies for coarse-to-fine semantic segmentation," *Technologies*, vol. 8, no. 1, p. 1, 2020.

[25] U. Michieli, E. Borsato, L. Rossi, and P. Zanuttigh, "GMNet: Graph Matching Network for Large Scale Part Semantic Segmentation in the Wild," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[26] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[27] U. Michieli, M. Biasetton, G. Agresti, and P. Zanuttigh, "Adversarial learning and self-teaching techniques for domain adaptation in semantic segmentation," *IEEE Transaction on Intelligent Vehicles*, 2020.

[28] T. Spadotto, M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation with multiple domain discriminators and adaptive self-training," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2020.

[29] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[30] M. Toldo, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised domain adaptation for mobile semantic segmentation based on cycle consistency and feature alignment," *Image and Vision Computing*, 2020.

[31] M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation via orthogonal and clustered embeddings," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[32] F. Barbato, M. Toldo, U. Michieli, and P. Zanuttigh, "Latent space regularization for unsupervised domain adaptation in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[33] F. Barbato, U. Michieli, M. Toldo, and P. Zanuttigh, "Adapting segmentation networks to new domains by disentangling latent representations," *arXiv preprint arXiv:2108.03021*, 2021.

[34] U. Michieli and M. Ozay, "Are All Users Treated Fairly in Federated Learning Systems?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[35] ——, "Prototype guided federated learning of visual feature representations," *arXiv preprint arXiv:2105.08982*, 2021.

[36] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[37] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[38] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[39] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[40] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 1996, pp. 640–646.

[41] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, part C (Applications and Reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[42] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2001, pp. 409–415.

[43] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[44] S. Doncieux, N. Bredeche, L. L. Goff, B. Girard, A. Coninx, O. Sigaud, M. Khamassi, N. Díaz-Rodríguez, D. Filliat, T. Hospedales *et al.*, "Dream architecture: a developmental approach to open-ended learning in robotics," *arXiv preprint arXiv:2005.06223*, 2020.

[45] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[46] S. Grossberg, "Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world," *Neural Networks*, vol. 37, pp. 1–47, 2013.

[47] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[48] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2001–2010.

[49] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 12, pp. 2935–2947, 2018.

[50] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.

[51] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 3400–3409.

[52] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. Heck, "Efficient incremental learning for mobile object detection," *arXiv preprint arXiv:1904.00781*, 2019.

[53] F. Ozdemir and O. Goksel, "Extending pretrained segmentation networks with additional anatomical structures," *International journal of computer assisted radiology and surgery*, vol. 14, no. 7, pp. 1187–1195, 2019.

[54] O. Tasar, Y. Tarabalka, and P. Alliez, "Incremental learning for semantic segmentation of large-scale remote sensing data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3524–3537, 2019.

[55] M. Klingner, A. Bär, P. Donn, and T. Fingscheidt, "Class-incremental learning for semantic segmentation re-using neither old data nor old labels," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[56] A. Douillard, Y. Chen, A. Dapogny, and M. Cord, "Plop: Learning without forgetting for continual semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[57] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, N. Díaz-Rodríguez, and D. Filliat, "Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer," in *Proceedings of the International Conference on Machine Learning Workshops (ICMLW)*, 2019.

[58] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, G. Cai, N. Díaz-Rodríguez, and D. Filliat, "Discorl: Continual reinforcement learning via policy distillation," *arXiv preprint arXiv:1907.05855*, 2019.

[59] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[60] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "Continual learning: A comparative study on how to defy forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[61] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[62] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 177–186.

[63] Y.-X. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2471–2480.

[64] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[65] D. Roy, P. Panda, and K. Roy, "Tree-CNN: a hierarchical deep convolutional neural network for incremental learning," *Neural Networks*, 2019.

[66] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017, pp. 3987–3995.

[67] T. Lesort, A. Stoian, and D. Filliat, "Regularization shortcomings for continual learning," *arXiv preprint arXiv:1912.03049*, 2019.

[68] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[69] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[70] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[71] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu, "Incremental classifier learning with generative adversarial networks," *arXiv preprint arXiv:1802.00853*, 2018.

[72] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, and M. Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 321–11 329.

[73] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 2990–2999.

[74] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 831–839.

[75] T. Lesort, "Continual learning: Tackling catastrophic forgetting in deep neural networks with replay processes," *arXiv preprint arXiv:2007.00487*, 2020.

[76] R. Istrate, A. C. I. Malossi, C. Bekas, and D. Nikolopoulos, "Incremental training of deep convolutional neural networks," *arXiv preprint arXiv:1803.10232*, 2018.

[77] S. S. Sarwar, A. Ankit, and K. Roy, "Incremental learning in deep convolutional neural networks using partial network sharing," *IEEE Access*, 2017.

[78] X. Dai, H. Yin, and N. K. Jha, "Incremental learning using a grow-and-prune paradigm with efficient neural networks," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[79] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 114, no. 13, pp. 3521–3526, 2017.

[80] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[81] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5138–5146.

[82] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. S. Davis, "M2kd: Multi-model and multi-level knowledge distillation for incremental learning," *Proceedings of the British Machine Vision Conference (BMVC)*, 2020.

[83] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan, "Active long term memory networks," *arXiv preprint arXiv:1606.02355*, 2016.

[84] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[85] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1365–1374.

[86] N. Passalis, M. Tzelepi, and A. Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2339–2348.

[87] L. Yu, V. O. Yazici, X. Liu, J. v. d. Weijer, Y. Cheng, and A. Ramisa, "Learning metrics from teachers: Compact networks for image embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2907–2916.

[88] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82.

[89] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1717–1724.

[90] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[91] J. Mańdziuk and L. Shastri, "Incremental class learning approach and its application to handwritten digit recognition," *Information Sciences*, vol. 141, no. 3-4, pp. 193–217, 2002.

[92] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," *arXiv preprint arXiv:1607.00122*, 2016.

[93] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[94] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 532–547.

[95] S. Hou, X. Pan, C. Change Loy, Z. Wang, and D. Lin, "Lifelong learning via progressive distillation and retrospection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 437–452.

[96] M. Welling, "Herding dynamical weights to learn," in *Proceedings of the Annual International Conference on Machine Learning (ICML)*. ACM, 2009.

[97] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 11 849–11 860.

[98] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. Heck, "Rilod: near real-time incremental learning for object detection at the edge," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 113–126.

[99] A. Douillard, Y. Chen, A. Dapogny, and M. Cord, "Tackling catastrophic forgetting and background shift in continual semantic segmentation," *arXiv preprint arXiv:2106.15287*, 2021.

[100] S. Cha, B. Kim, Y. Yoo, and T. Moon, "Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning," *arXiv preprint arXiv:2106.11562*, 2021.

[101] G. Nguyen, S. Chen, T. Do, T. J. Jun, H.-J. Choi, and D. Kim, "Dissecting catastrophic forgetting in continual learning by deep visualization," *arXiv preprint arXiv:2001.01578*, 2020.

[102] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 35, no. 8, pp. 1798–1828, 2013.

[103] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[104] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 69–77.

[105] X. Peng, Z. Huang, X. Sun, and K. Saenko, "Domain agnostic learning with disentangled representations," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 5102–5112.

[106] A. Achille, T. Eccles, L. Matthey, C. Burgess, N. Watters, A. Lerchner, and I. Higgins, "Life-long disentangled representation learning with cross-domain latent homologies," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[107] K. Javed and M. White, "Meta-learning representations for continual learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[108] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[109] T. J. Draelos, N. E. Miner, C. C. Lamb, J. A. Cox, C. M. Vineyard, K. D. Carlson, W. M. Severa, C. D. James, and J. B. Aimone, "Neurogenesis deep learning: Extending deep networks to accommodate new classes," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 526–533.

[110] N. Kamra, U. Gupta, and Y. Liu, "Deep generative dual memory network for continual learning," *arXiv preprint arXiv:1710.10368*, 2017.

[111] Z. Huang, W. Hao, X. Wang, M. Tao, J. Huang, W. Liu, and X.-S. Hua, "Half-real half-fake distillation for class-incremental semantic segmentation," *arXiv preprint arXiv:2104.00875*, 2021.

[112] Q. Hou, M.-M. Cheng, J. Liu, and P. H. Torr, "Webseg: Learning semantic segmentation from web searches," *arXiv preprint arXiv:1803.09859*, 2018.

[113] D. Modolo and V. Ferrari, "Learning semantic part-based models from google images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 6, pp. 1502–1509, 2017.

[114] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.

[115] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision (IJCV)*, vol. 81, no. 1, pp. 2–23, 2009.

[116] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[117] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 633–641.

[118] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, "Don't forget, there is more than forgetting: new metrics for continual learning," in *Advances in Neural Information Processing Systems Workshops (NeurIPSW)*, 2018.

[119] V. Nekrasov, "Pre-computed weights for ResNet-101," https://github.com/DrSleep/tensorflow-deeplab-resnet, Accessed: 2020-03.

[120] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[121] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[122] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[123] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 1321–1330.

[124] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, "What is a good evaluation measure for semantic segmentation?" in *Proceedings of the British Machine Vision Conference (BMVC)*, vol. 27, 2013, p. 2013.

[125] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[126] R. Aljundi, M. Rohrbach, and T. Tuytelaars, "Selfless sequential learning," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[127] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[128] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3733–3742.

[129] C. Zhuang, A. L. Zhai, and D. Yamins, "Local aggregation for unsupervised learning of visual embeddings," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 6002–6012.

[130] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," *arXiv preprint arXiv:1906.05849*, 2019.

[131] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9729–9738.

[132] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6707–6717.

[133] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[134] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 2051–2060.

[135] M. Ye, X. Zhang, P. C. Yuen, and S.-F. Chang, "Unsupervised embedding learning via invariant and spreading instance feature," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6210–6219.

[136] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 9865–9874.

[137] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[138] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4893–4902.

[139] J. Liang, R. He, Z. Sun, and T. Tan, "Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2975–2984.

[140] N. Dong and E. P. Xing, "Few-shot semantic segmentation with prototype learning." in *Proceedings of the British Machine Vision Conference (BMVC)*, vol. 3, 2018.

[141] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "Panet: Few-shot image semantic segmentation with prototype alignment," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 9197–9206.

[142] Z. Tian, X. Lai, L. Jiang, M. Shu, H. Zhao, and J. Jia, "Generalized few-shot semantic segmentation," *arXiv preprint arXiv:2010.05210*, 2020.

[143] P. O. Pinheiro, "Unsupervised domain adaptation with similarity learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8004–8013.

[144] S. Wu, J. Zhong, W. Cao, R. Li, Z. Yu, and H.-S. Wong, "Improving domain-specific classification by collaborative learning with adaptation networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5450–5457.

[145] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[146] Z. Deng, Y. Luo, and J. Zhu, "Cluster alignment with a teacher for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 9944–9953.

[147] S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa, "Generalized domain-adaptive dictionaries," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 361–368.

[148] H. Zhang, V. M. Patel, S. Shekhar, and R. Chellappa, "Domain adaptive sparse representation-based classification," in *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1. IEEE, 2015, pp. 1–8.

[149] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4077–4087.

[150] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, pp. 721–731, 2018.

[151] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2517–2526.

[152] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[153] C. He, R. Wang, S. Shan, and X. Chen, "Exemplar-supported generative reproduction for class incremental learning." in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018, p. 98.

[154] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 1431–1439.

[155] S. K. Divvala, A. Farhadi, and C. Guestrin, "Learning everything about anything: Webly-supervised visual concept learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3270–3277.

[156] L. Niu, A. Veeraraghavan, and A. Sabharwal, "Webly supervised learning meets zero-shot learning: A hybrid approach for fine-grained classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7171–7180.

[157] B. Jin, M. V. Ortiz Segovia, and S. Susstrunk, "Webly supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3626–3635.

[158] T. Shen, G. Lin, C. Shen, and I. Reid, "Bootstrapping the performance of webly supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1363–1371.

[159] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han, "Weakly supervised semantic segmentation using web-crawled videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7322–7330.

[160] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[161] "Tensorflow module of BigGAN-deep 512, https://tfhub.dev/deepmind/biggan-deep-512/1. Accessed on 18/03/2020."

[162] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[163] "TensorFlow module of EfficientNet-b2, https://tfhub.dev/google/efficientnet/b2/classification/1. Accessed on 18/03/2020."

[164] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.

[165] O. Stretcu, E. A. Platanios, T. M. Mitchell, and B. Póczos, "Coarse-to-fine curriculum learning," *arXiv preprint arXiv:2106.04072*, 2021.

[166] J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.

[167] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[168] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[169] C. Couprie, C. Farabet, L. Najman, and Y. Lecun, "Convolutional nets and watershed cuts for real-time semantic labeling of RGBD videos." *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 3489–3511, 2014.

[170] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 2650–2658, 2015.

[171] J. Wang, Z. Wang, D. Tao, S. See, and G. Wang, "Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 664–679, 2016.

[172] U. Michieli, M. Camporese, A. Agiollo, G. Pagnutti, and P. Zanuttigh, "Region Merging Driven by Deep Learning for RGB-D Segmentation and Labeling," *International Conference on Distributed Smart Cameras (ICDSC), Trento (Italy)*, 2019.

[173] X. Ren, L. Bo, and D. Fox, "RGB-D scene labeling: Features and algorithms," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[174] Y. Wang, B. Liang, M. Ding, and J. Li, "Dense semantic labeling with atrous spatial pyramid pooling and decoder for high-resolution remote sensing imagery," *Remote Sensing*, vol. 11, no. 1, p. 20, 2019.

[175] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 345–360, 2014.

[176] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3150–3158, 2016.

[177] Z. Ren and Y. Jae Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 762–771.

[178] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2018.

[179] E. Zakirov, "Pre-computed weights for Xception," https://github.com/bonlime/keras-deeplab-v3-plus/, Accessed: 2019-10-20.

[180] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[181] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[182] G. Pagnutti, L. Minto, and P. Zanuttigh, "Segmentation and semantic labelling of rgbd data with convolutional neural networks and surface fitting," *IET Computer Vision*, vol. 11, no. 8, pp. 633–642, 2017.

[183] A. C. Măller and S. Behnke, "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images." *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[184] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation," *International Journal of Computer Vision (IJCV)*, vol. 112, no. 2, pp. 133–149, 2015.

[185] C. Cadena and J. Košecka, "Semantic parsing for priming object detection in indoors rgb-d scenes," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 582–597, 2015.

[186] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from rgb-d video," *Journal of Real-Time Image Processing*, vol. 10, pp. 599–609, 2013.

[187] A. Wang, J. Lu, G. Wang, J. Cai, and T. Cham, "Multi-modal unsupervised feature learning for RGB-D scene labeling," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 453–467, 2014.

[188] A. Hermans, G. Floros, and B. Leibe, "Dense 3D semantic mapping of indoor scenes from rgb-d images," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2631–2638, 2014.

[189] S. Khan, M. Bennamoun, F. Sohel, and R. Togneri, "Geometry driven semantic labeling of indoor scenes," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 679–694, 2014.

[190] D. Lin, C. Guangyong, D. Cohen-Or, P.-A. Heng, and H. Huang, "Cascaded feature network for semantic segmentation of rgb-d images," *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017-10.

[191] H. Liu, W. Wu, and X. Wang, "Rgb-d joint modelling with scene geometric information for indoor semantic segmentation," *Multimedia Tools and Applications*, 2018-05.

[192] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3d graph neural networks for rgbd semantic segmentation," *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 5209–5218, 2017-10.

[193] J. Dong, Q. Chen, X. Shen, J. Yang, and S. Yan, "Towards unified human parsing and pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 843–850.

[194] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1385–1392.

[195] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1971–1978.

[196] H. Azizpour and I. Laptev, "Object detection using strongly-supervised deformable part models," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 836–849.

[197] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 834–849.

[198] Y. Wang, D. Tran, Z. Liao, and D. Forsyth, "Discriminative hierarchical part-based models for human parsing and action recognition," *Journal of Machine Learning Research*, vol. 13, no. Oct, pp. 3075–3102, 2012.

[199] J. Sun and J. Ponce, "Learning discriminative part detectors for image classification and cosegmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013, pp. 3400–3407.

[200] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5546–5555.

[201] X. Liang, S. Liu, X. Shen, J. Yang, L. Liu, J. Dong, L. Lin, and S. Yan, "Deep human parsing with active template regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 37, no. 12, pp. 2402–2414, 2015.

[202] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, "Parsing clothing in fashion photographs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3570–3577.

[203] L. L. Zhu, Y. Chen, C. Lin, and A. Yuille, "Max margin learning of hierarchical configural deformable templates (hcdts) for efficient object parsing and pose estimation," *International Journal of Computer Vision (IJCV)*, vol. 93, no. 1, pp. 1–21, 2011.

[204] S. Eslami and C. Williams, "A generative model for parts-based object segmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 100–107.

[205] Y. Song, X. Chen, J. Li, and Q. Zhao, "Embedding 3d geometric features for rigid object part segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 580–588.

[206] W. Lu, X. Lian, and A. Yuille, "Parsing semantic parts of cars using graphical models and segment appearance consistency," *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[207] J. Wang and A. L. Yuille, "Semantic part segmentation using compositional model combining shape and appearance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1788–1797.

[208] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Joint object and part segmentation using deep learned potentials," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 1573–1581.

[209] H. Haggag, A. Abobakr, M. Hossny, and S. Nahavandi, "Semantic body parts segmentation for quadrupedal animals," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 000 855–000 860.

[210] Y. Zhao, J. Li, Y. Zhang, and Y. Tian, "Multi-class part parsing with joint boundary-semantic awareness," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 9177–9186.

[211] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 447–456.

[212] F. Xia, P. Wang, X. Chen, and A. L. Yuille, "Joint multi-person pose estimation and semantic part segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6769–6778.

[213] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille, "Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 648–663.

[214] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 12, pp. 2481–2495, 2017.

[215] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3640–3649.

[216] X. Liang, K. Gong, X. Shen, and L. Lin, "Look into person: Joint body parsing & pose estimation network and a new benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 41, no. 4, pp. 871–885, 2018.

[217] X. Liang, L. Lin, X. Shen, J. Feng, S. Yan, and E. P. Xing, "Interpretable structure-evolving lstm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1010–1019.

[218] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 125–143.

[219] H.-S. Fang, G. Lu, X. Fang, J. Xie, Y.-W. Tai, and C. Lu, "Weakly and semi supervised human body part parsing via pose-guided knowledge transfer," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[220] X. Nie, J. Feng, and S. Yan, "Mutual learning to adapt for joint human parsing and pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 502–517.

[221] J. Zhao, J. Li, X. Nie, F. Zhao, Y. Chen, Z. Wang, J. Feng, and S. Yan, "Self-supervised neural aggregation networks for human parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 7–15.

[222] F. Xia, J. Zhu, P. Wang, and A. Yuille, "Pose-guided human parsing with deep learned features," *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[223] N. Díaz-Rodríguez, A. Lamas, J. Sanchez, G. Franchi, I. Donadello, S. Tabik, D. Filliat, P. Cruz, R. Montes, and F. Herrera, "Explainable neural-symbolic learning (x-nesyl) methodology to fuse deep learning representations with expert knowledge graphs: the monumai cultural heritage use case," *Information Fusion*, 2021.

[224] A. Bennetot, J.-L. Laurent, R. Chatila, and N. Díaz-Rodríguez, "Towards explainable neural-symbolic visual reasoning," *arXiv preprint arXiv:1909.09065*, 2019.

[225] F. Emmert-Streib, M. Dehmer, and Y. Shi, "Fifty years of graph matching, network alignment and network comparison," *Information Sciences*, vol. 346, pp. 180–197, 2016.

[226] L. Livi and A. Rizzi, "The graph matching problem," *Pattern Analysis and Applications*, vol. 16, no. 3, pp. 253–283, 2013.

[227] D. Das and C. G. Lee, "Unsupervised domain adaptation using regularized hyper-graph matching," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3758–3762.

[228] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari, "Do semantic parts emerge in convolutional neural networks?" *International Journal of Computer Vision (IJCV)*, vol. 126, no. 5, pp. 476–494, 2018.

[229] L.-C. Chen, "DeepLab official TensorFlow implementation," Accessed: 2020-03-01. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/deeplab

[230] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[231] S. Sun, H. Shi, and Y. Wu, "A survey of multi-source domain adaptation," *Information Fusion*, vol. 24, pp. 84–92, 2015.

[232] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *Domain Adaptation in Computer Vision Application*, 2017.

[233] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *Proc. of the 45th annual meeting of the association of computational linguistics*, 2007, pp. 264–271.

[234] F. Fang, K. Dutta, and A. Datta, "Domain adaptation for sentiment classification in light of multiple sources," *INFORMS Journal on Computing*, vol. 26, no. 3, pp. 586–598, 2014.

[235] J. Jiang, "A literature survey on domain adaptation of statistical classifiers," *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, vol. 3, pp. 1–12, 2008.

[236] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[237] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015, pp. 97–105.

[238] P. P. Busto and J. Gall, "Open set domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 754–763.

[239] J. Zhuo, S. Wang, S. Cui, and Q. Huang, "Unsupervised open domain recognition by semantic discrepancy minimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 750–759.

[240] J. N. Kundu, R. M. Venkatesh, N. Venkat, A. Revanur, and R. V. Babu, "Class-incremental domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[241] H. T. Ho and R. Gopalan, "Model-driven domain adaptation on product manifolds for unconstrained face recognition," *International Journal of Computer Vision (IJCV)*, vol. 109, no. 1-2, pp. 110–125, 2014.

[242] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2010, pp. 213–226.

[243] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proceedings of the European Conference on Computer Vision (ECCV)*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[244] K. Saito, D. Kim, S. Sclaroff, and K. Saenko, "Universal domain adaptation through self supervision," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[245] M. Bucher, T.-H. Vu, M. Cord, and P. Pérez, "Buda: Boundless unsupervised domain adaptation in semantic segmentation," *arXiv preprint arXiv:2004.01130*, 2020.

[246] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang, "Weakly supervised object localization with progressive domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3512–3520.

[247] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5001–5009.

[248] F. Sun and W. Li, "Saliency guided deep network for weakly-supervised image segmentation," *Pattern Recognition Letters (PRL)*, 2019.

[249] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5715–5725.

[250] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[251] A. Vezhnevets and J. M. Buhmann, "Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 3249–3256.

[252] D. Pathak, E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional multi-class multiple instance learning," *arXiv preprint arXiv:1412.7144*, 2014.

[253] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 1742–1750.

[254] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 1796–1804.

[255] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, J. Feng, Y. Zhao, and S. Yan, "STC: A simple to complex framework for weakly-supervised semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 11, pp. 2314–2320, 2017.

[256] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 1495–1503.

[257] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 1635–1643.

[258] N. Souly, C. Spampinato, and M. Shah, "Semi and weakly supervised semantic segmentation using generative adversarial network," *arXiv preprint arXiv:1703.09695*, 2017.

[259] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 695–711.

[260] Z. Huang, X. Wang, J. Wang, W. Liu, and J. Wang, "Weakly-supervised semantic segmentation network with deep seeded region growing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7014–7023.

[261] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1568–1576.

[262] J. Ahn and S. Kwak, "Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4981–4990.

[263] J. Lee, E. Kim, S. Lee, J. Lee, and S. Yoon, "Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5267–5276.

[264] J. Ahn, S. Cho, and S. Kwak, "Weakly supervised learning of instance segmentation with inter-pixel relations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2209–2218.

[265] P. Z. Ramirez, A. Tonioni, S. Salti, and L. D. Stefano, "Learning across tasks and domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[266] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[267] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[268] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016, pp. 2058–2065.

[269] B. Sun and K. Saenko, "Deep CORAL: correlation alignment for deep domain adaptation," in *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2016, pp. 443–450.

[270] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015, pp. 1180–1189.

[271] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 1, pp. 2096–2030, 2016.

[272] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7167–7176.

[273] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "FCNs in the wild: Pixel-level adversarial and constraint-based adaptation," *arXiv preprint arXiv:1612.02649*, 2016.

[274] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional learning for domain adaptation of semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[275] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[276] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang, "Crdoco: Pixel-level domain transfer with cross-domain consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[277] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. Frank Wang, and M. Sun, "No more discrimination: Cross city adaptation of road scene segmenters," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 1992–2001.

[278] L. Du, J. Tan, H. Yang, J. Feng, X. Xue, Q. Zheng, X. Ye, and X. Zhang, "SSF-DAN: separated semantic feature based domain adaptation network for semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[279] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3752–3761.

[280] Z. Murez, S. Kolouri, D. J. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[281] X. Zhu, H. Zhou, C. Yang, J. Shi, and D. Lin, "Penalizing top performers: Conservative loss for semantic segmentation adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 568–583.

[282] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7472–7481.

[283] Y. Chen, W. Li, X. Chen, and L. V. Gool, "Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1841–1850.

[284] W. Chang, H. Wang, W. Peng, and W. Chiu, "All about structure: Adapting structural information across domains for boosting semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1900–1909.

[285] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[286] J. Yang, W. An, S. Wang, X. Zhu, C. Yan, and J. Huang, "Label-driven reconstruction for domain adaptation in semantic segmentation," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[287] Y. Chen, W. Li, and L. Van Gool, "Road: Reality oriented adaptation for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7892–7901.

[288] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, "Fully convolutional adaptation networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6810–6818.

[289] H. Huang, Q. Huang, and P. Krähenbühl, "Domain transfer through deep activation matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[290] Y. Luo, P. Liu, T. Guan, J. Yu, and Y. Yang, "Significance-aware information bottleneck for domain adaptive semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[291] T. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "DADA: depth-aware domain adaptation in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 7363–7372.

[292] Y.-H. Tsai, K. Sohn, S. Schulter, and M. Chandraker, "Domain adaptation for structured output via discriminative patch representations," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1456–1465.

[293] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2242–2251.

[294] X. Peng and K. Saenko, "Synthetic to real adaptation with generative correlation alignment networks," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1982–1991.

[295] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 318–335.

[296] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 597–613.

[297] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *arXiv preprint arXiv:1602.05110*, 2016.

[298] Q. Zhou, Z. Feng, G. Cheng, X. Tan, J. Shi, and L. Ma, "Uncertainty-aware consistency regularization for cross-domain semantic segmentation," *arXiv preprint arXiv:2004.08878*, 2020.

[299] C. Qin, L. Wang, Y. Zhang, and Y. Fu, "Generatively inferential co-training for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2019, pp. 1055–1064.

[300] P. Li, X. Liang, D. Jia, and E. P. Xing, "Semantic-aware grad-gan for virtual-to-real urban scene adaption," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[301] Y. Yang, D. Lao, G. Sundaramoorthi, and S. Soatto, "Phase consistent ecological domain adaptation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[302] R. Gong, W. Li, Y. Chen, and L. V. Gool, "DLOW: domain flow for adaptation and generalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2477–2486.

[303] F. Ragusa, D. Di Mauro, A. Palermo, A. Furnari, and G. M. Farinella, "Semantic object segmentation in cultural sites using real and synthetic data," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 1964–1971.

[304] K. Lee, G. Ros, J. Li, and A. Gaidon, "SPIGAN: privileged adversarial learning from simulation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[305] J. Choi, T. Kim, and C. Kim, "Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 6830–6840.

[306] W. Hong, Z. Wang, M. Yang, and J. Yuan, "Conditional generative adversarial network for structured domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1335–1344.

[307] F. Pizzati, R. d. Charette, M. Zaccaria, and P. Cerri, "Domain bridge for unpaired image-to-image translation and unsupervised domain adaptation," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 2990–2998.

[308] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 179–196.

[309] Z. Wu, X. Han, Y. Lin, M. G. Uzunbas, T. Goldstein, S. Lim, and L. S. Davis, "DCAN: dual channel-wise alignment networks for unsupervised scene adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 535–552.

[310] Z. Wu, X. Wang, J. Gonzalez, T. Goldstein, and L. Davis, "ACE: adapting to changing environments for semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 2121–2130.

[311] A. Dundar, M. Liu, T. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation," *arXiv preprint arXiv:1807.09384*, 2018.

[312] Y. Yang and S. Soatto, "FDA: fourier domain adaptation for semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[313] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 262–270.

[314] ——, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

[315] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 1510–1519.

[316] F. Pizzati, P. Cerri, and R. de Charette, "Comogan: continuous model-guided image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 14 288–14 298.

[317] B. Lütjens, B. Leshchinskiy, C. Requena-Mesa, F. Chishtie, N. Díaz-Rodríguez, O. Boulais, A. Sankaranarayanan, A. Piña, Y. Gal, C. Raïssi *et al.*, "Physically-consistent generative adversarial networks for coastal flood visualization," *arXiv preprint arXiv:2104.04785*, 2021.

[318] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[319] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3723–3732.

[320] K. Watanabe, K. Saito, Y. Ushiku, and T. Harada, "Multichannel semantic segmentation with unsupervised domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[321] S. Lee, D. Kim, N. Kim, and S.-G. Jeong, "Drop to adapt: Learning discriminative features for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 91–100.

[322] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Actes de CAP 05, Conférence francophone sur l'apprentissage automatique*, 2005, pp. 281–296.

[323] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 289–305.

[324] Y. Zou, Z. Yu, X. Liu, B. V. Kumar, and J. Wang, "Confidence regularized self-training," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 5982–5991.

[325] M. Chen, H. Xue, and D. Cai, "Domain adaptation for semantic segmentation with maximum squares loss," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[326] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 2020–2030.

[327] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[328] C. Sakaridis, D. Dai, S. Hecker, and L. Van Gool, "Model adaptation with synthetic and real data for semantic dense foggy scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 687–704.

[329] D. Dai, C. Sakaridis, S. Hecker, and L. Van Gool, "Curriculum model adaptation with synthetic and real data for semantic foggy scene understanding," *International Journal of Computer Vision (IJCV)*, pp. 1–23, 2019.

[330] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1106–1114.

[331] Q. Lian, F. Lv, L. Duan, and B. Gong, "Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 6758–6767.

[332] L. Tian, Y. Tang, L. Hu, Z. Ren, and W. Zhang, "Domain adaptation by class centroid matching and local manifold self-learning," *IEEE Transactions on Image Processing (TIP)*, 2020.

[333] Q. Wang and T. P. Breckon, "Unsupervised domain adaptation via structured prediction based selective pseudo-labeling," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 6243–6250.

[334] H. Tang, K. Chen, and K. Jia, "Unsupervised domain adaptation via structurally regularized deep clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8722–8732.

[335] W. Shi, Y. Gong, D. Cheng, X. Tao, and N. Zheng, "Entropy and orthogonality based deep discriminative feature learning for object recognition," *Pattern Recognition*, vol. 81, pp. 71–80, 2018.

[336] W. Wang, D. Yang, F. Chen, Y. Pang, S. Huang, and Y. Ge, "Clustering with orthogonal autoencoder," *IEEE Access*, vol. 7, pp. 62 421–62 432, 2019.

[337] H. Choi, A. Som, and P. Turaga, "Role of orthogonality constraints in improving properties of deep networks for image classification," *arXiv preprint arXiv:2009.10762*, 2020.

[338] M. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2007, pp. 1137–1144.

[339] M. Bucher, T. Vu, M. Cord, and P. Pérez, "Zero-shot semantic segmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 466–477.

[340] D. Smith and B. Burke, "Gartner's 2019 hype cycle for emerging technologies," *Gartner*, 2019.

[341] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.

[342] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[343] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The Mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 4990–4999.

[344] D. Di Mauro, A. Furnari, G. Patanè, S. Battiato, and G. M. Farinella, "Sceneadapt: Scene-based domain adaptation for semantic segmentation using adversarial learning," *Pattern Recognition Letters (PRL)*, vol. 136, pp. 175–182, 2020.

[345] U. Michieli and L. Badia, "Game theoretic analysis of road user safety scenarios involving autonomous vehicles," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2018, pp. 1377–1381.

[346] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[347] J. Kim and C. Park, "Attribute dissection of urban road scenes for efficient dataset integration," in *International Joint Conference on Artificial Intelligence Workshops*, 2018, pp. 8–15.

[348] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[349] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 700–708.

[350] W.-C. Hung, Y.-H. Tsai, Y.-T. Liou34, Y.-Y. Lin, and M.-H. Yang15, "Adversarial learning for semi-supervised semantic segmentation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[351] X. Liu, J. Cao, T. Fu, Z. Pan, W. Hu, K. Zhang, and J. Liu, "Semi-supervised automatic segmentation of layer and fluid region in retinal optical coherence tomography images using adversarial learning," *IEEE Access*, vol. 7, pp. 3046–3061, 2019.

[352] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," in *NIPS Workshop on Adversarial Training*, 2016.

[353] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars, "A deeper look at dataset bias," in *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 37–55.

[354] A. Torralba and A. Efros, "Unbiased look at dataset bias," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2011, pp. 1521–1528.

[355] B. Gong, F. Sha, and K. Grauman, "Overcoming dataset bias: An unsupervised domain adaptation approach," in *NIPS Workshop on Large Scale Visual Recognition and Retrieval*, vol. 3. Citeseer, 2012.

[356] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 158–171.

[357] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 16, no. 6, pp. 641–647, 1994.

[358] G. Song, H. Myeong, and K. Mu Lee, "Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1760–1768.

[359] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8503–8512.

[360] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[361] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[362] "Pre-trained weights for MobileNet-v2 on Pascal VOC 2012 dataset, https://github.com/tensorflow/models/tree/master/research/deeplab. Accessed on 20/12/2019."

[363] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[364] "Implementation on TensorFlow of the CycleGAN framework, https://github.com/vanhuyz/CycleGAN-TensorFlow. Accessed on 20/12/2019."

[365] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[366] C. Li, D. Du, L. Zhang, L. Wen, T. Luo, Y. Wu, and P. Zhu, "Spatial attention pyramid network for unsupervised domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[367] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[368] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[369] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson, "Dacs: Domain adaptation via cross-domain mixed sampling," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1379–1389.

[370] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and surface variations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[371] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *Conference of Machine Learning and Systems (MLSys)*, 2019.

[372] P. Kairouz and H. B. McMahan, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, 2021.

[373] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[374] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[375] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Conference on Machine Learning and Systems (MLSys)*, 2020.

[376] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[377] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[378] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized federated learning: An attentive collaboration approach," *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[379] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[380] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[381] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[382] P. Yu and Y. Liu, "Federated object detection: Optimizing object detection model with federated learning," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ser. ICVISP 2019. New York, NY, USA: Association for Computing Machinery, 2019.

[383] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[384] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 3973–3983.

[385] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[386] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5132–5143.

[387] C. Zhang and J. A. Shah, "Fairness in multi-agent sequential decision-making," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2636–2644.

[388] J. Jiang and Z. Lu, "Learning fairness in multi-agent systems," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[389] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 4615–4625.

[390] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[391] J. Zhang, C. Li, A. Robles-Kelly, and M. Kankanhalli, "Hierarchically fair federated learning," *arXiv preprint arXiv:2004.10386*, 2020.

[392] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[393] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[394] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2014, pp. 1000–1008.

[395] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[396] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[397] W. Shakespeare, *The complete works of William Shakespeare*. Publically available at https://www.gutenberg.org/ebooks/100.

[398] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015-12.

[399] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[400] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: a Retrospective," *International Journal of Computer Vision (IJCV)*, vol. 111, no. 1, pp. 98–136, 2015.

[401] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[402] M. Rapp, R. Khalili, and J. Henkel, "Distributed learning on heterogeneous resource-constrained devices," *arXiv preprint arXiv:2006.05403*, 2020.

[403] C. Wang, Y. Yang, and P. Zhou, "Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 394–410, 2020.

[404] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[405] K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum, "Infinite mixture prototypes for few-shot learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 232–241.

[406] F. Cermelli, M. Mancini, Y. Xian, Z. Akata, and B. Caputo, "A few guidelines for incremental few-shot segmentation," *arXiv preprint arXiv:2012.01415*, 2020.

[407] J. Kim, T.-H. Oh, S. Lee, F. Pan, and I. S. Kweon, "Variational prototyping-encoder: One-shot learning with prototypical images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9462–9470.

[408] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang, "Boosting few-shot learning with adaptive margin loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 576–12 584.

[409] X. Liu, F. Zhou, J. Liu, and L. Jiang, "Meta-learning based prototype-relation network for few-shot classification," *Neurocomputing*, vol. 383, pp. 224–234, 2020.

[410] Y. Pan, T. Yao, Y. Li, Y. Wang, C. Ngo, and T. Mei, "Transferrable prototypical networks for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2239–2247.

[411] M. Zhang, T. Wang, J. H. Lim, G. Kreiman, and J. Feng, "Variational prototype replays for continual learning," *arXiv preprint arXiv:1905.09447*, 2019.

[412] B. Hammer, M. Strickert, and T. Villmann, "On the generalization ability of grlvq networks," *Neural Processing Letters*, vol. 21, no. 2, pp. 109–120, 2005.

[413] X.-B. Jin, C.-L. Liu, and X. Hou, "Regularized margin-based conditional log-likelihood loss for prototype learning," *Pattern Recognition*, vol. 43, no. 7, pp. 2428–2438, 2010.

[414] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," *Neural Computing and Applications*, vol. 25, no. 3, pp. 511–524, 2014.

[415] A. Sato and K. Yamada, "Generalized learning vector quantization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 1995, pp. 423–429.

[416] P. Schneider, M. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization," *Neural computation*, vol. 21, no. 12, pp. 3532–3561, 2009.

[417] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin analysis of the lvq algorithm," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 2, 2002, pp. 462–469.

[418] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi, and J. Uijlings, "The devil is in the decoder: Classification, regression and gans," in *International Journal of Computer Vision (IJCV)*, vol. 127, no. 11. Springer, 2019, pp. 1694–1706.

[419] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1426–1435.

[420] S. Rosset, J. Zhu, and T. Hastie, "Margin maximizing loss functions." in *Advances in Neural Information Processing Systems (NeurIPS)*, 2003, pp. 1237–1244.

[421] N. Rodríguez-Barroso, G. Stipcich, D. Jiménez-López, J. A. Ruiz-Millán, E. Martínez-Cámara, G. González-Seco, M. V. Luzón, M. A. Veganzones, and F. Herrera, "Federated learning and differential privacy: Software tools analysis, the sherpa.ai fl framework and methodological guidelines for preserving data privacy," *Information Fusion*, vol. 64, pp. 270–292, 2020.

[422] M. Ribero, J. Henderson, S. Williamson, and H. Vikalo, "Federating recommendations using differentially private prototypes," *arXiv preprint arXiv:2003.00602*, 2020.

[423] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems (NeurIPS)*. Cambridge, MA, USA: MIT Press, 1996, p. 134–140.

[424] M. J. Saberian and N. Vasconcelos, "Multiclass boosting: Theory and algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*. Red Hook, NY, USA: Curran Associates Inc., 2011, p. 2124–2132.

[425] M. Saberian and N. Vasconcelos, "Multiclass boosting: Margins, codewords, losses, and algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 20, no. 137, pp. 1–68, 2019. [Online]. Available: http://jmlr.org/papers/v20/17-137.html

[426] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin analysis of the lvq algorithm," in *Advances in Neural Information Processing Systems (NeurIPS)*. Cambridge, MA, USA: MIT Press, 2002, p. 479–486.

[427] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, 1984.

[428] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[429] W. Wan, J. Chen, T. Li, Y. Huang, J. Tian, C. Yu, and Y. Xue, "Information entropy based feature pooling for convolutional neural networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 3405–3414.

List of Publications

Journals

[430] F. Barbato, U. Michieli, M. Toldo, and P. Zanuttigh, "Adapting segmentation networks to new domains by disentangling latent representations," *submitted to IEEE Transactions on Multimedia*, 2021.

[431] U. Michieli and P. Zanuttigh, "Knowledge Distillation for Incremental Learning in Semantic Segmentation," *Elsevier Journal on Computer Vision and Image Understanding (CVIU)*, 2021.

[432] M. Toldo, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised Domain Adaptation for Mobile Semantic Segmentation based on Cycle Consistency and Feature Alignment," *Image and Vision Computing (IMAVIS)*, 2020.

[433] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised Domain Adaptation in Semantic Segmentation: a Review ," *Technologies*, vol. 8, no. 35, 2020.

[434] M. Mel, U. Michieli, and P. Zanuttigh, "Incremental and Multi-Task Learning Strategies for Coarse-to-Fine Semantic Segmentation," *Technologies, special issue on Computer Vision and Image Processing Technologies*, vol. 8, no. 1, 2020.

[435] U. Michieli, M. Biasetton, G. Agresti, and P. Zanuttigh, "Adversarial Learning and Self-Teaching Techniques for Domain Adaptation in Semantic Segmentation," *IEEE Transactions on Intelligent Vehicles (T-IV)*, vol. 5, no. 3, pp. 508–518, 2020.

Conference Proceedings

[436] A. Maracani*, U. Michieli*, M. Toldo*, and P. Zanuttigh, "RECALL: Replay-based Continual Learning in Semantic Segmentation," *International Conference on Computer Vision (ICCV) [acceptance rate=25.9%]*, 2021.

[437] U. Michieli and M. Ozay, "Are All Users Treated Fairly in Federated Learning Systems?" *Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Responsible Computer Vision (RCV)*, 2021.

[438] F. Barbato, M. Toldo, U. Michieli, and P. Zanuttigh, "Latent Space Regularization for Unsupervised Domain Adaptation in Semantic Segmentation," *Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Autonomous Driving (WAD)*, 2021.

[439] U. Michieli and P. Zanuttigh, "Continual Semantic Segmentation via Repulsion-Attraction of Sparse and Disentangled Latent Representations," *Computer Vision and Pattern Recognition (CVPR) 2021 [acceptance rate approx. 23.6%]*, 2021.

[440] M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised Domain Adaptation in Semantic Segmentation via Orthogonal and Clustered Embeddings," *Winter Conference on Applications of Computer Vision (WACV) [acceptance rate approx. 28%]*, 2021.

[441] U. Michieli, E. Borsato, L. Rossi, and P. Zanuttigh, "GMNet: Graph Matching Network for Large Scale Part Semantic Segmentation in the Wild," *European Conference on Computer Vision (ECCV), Glasgow (UK) [acceptance rate=26%]*, 2020.

[442] T. Spadotto, M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised Domain Adaptation with Multiple Domain Discriminators and Adaptive Self-Training," *International Conference on Pattern Recognition (ICPR), Milan (Italy) [first round acceptance rate=35.6%]*, 2020.

[443] U. Michieli and P. Zanuttigh, "Incremental Learning Techniques for Semantic Segmentation," *International Conference on Computer Vision (ICCV), Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV), Seoul (South Korea)*, 2019.

[444] U. Michieli, M. Camporese, A. Agiollo, G. Pagnutti, and P. Zanuttigh, "Region Merging Driven by Deep Learning for RGB-D Segmentation and Labeling," *International Conference on Distributed Smart Cameras (ICDSC), Trento (Italy)*, 2019.

[445] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes," *Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Autonomous Driving (WAD), Long Beach (US)*, 2019.

[446] U. Michieli and L. Badia, "Game Theoretic Analysis of Road User Safety Scenarios Involving Autonomous Vehicles," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Bologna (Italy)*, pp. 1377–1381, 2018.

[447] G. Cisotto, U. Michieli, and L. Badia, "A coherence study on EEG and EMG signals," *IEEE Global Wireless Summit (GWS), Aarhus (Denmark)*, pp. 372–376, 2016.

Book Chapters

[448] U. Michieli, M. Toldo, and P. Zanuttigh, "Unsupervised Domain Adaptation and Continual Learning in Semantic Segmentation," *Advanced Methods and Deep Learning in Computer Vision, Elsevier*, 2021.

Acknowledgments

The journey towards my Ph.D. has been one of the most exciting ones so far, but it would not have been possible without many people always believing in me, supporting me and contributing to this amazing goal. I would like to thank them all in these few lines.

First, I want to sincerely thank my advisor, Pietro Zanuttigh, who welcomed me in his lab and allowed me to grow as a researcher under his precious and assiduous supervision. Also, a special thank for trusting me as a teaching assistant for many consecutive years. My sincere gratitude goes to my advisor for the eight-months internship at Samsung Research UK, Mete Ozay, who relentelessly supervised me with passion and dedication growing my scientific perspectives. Thank you. The choice to pursue a Ph.D. goes back to my M.Sc. thesis project in Dresden, where Carlo Vittorio Cannistraci hosted me in his lab. I would like to thank him, Alessandro Muscoloni and all the other amazing colleagues for having transferred to me their passion and attitude towards research. Thank you!

I would like to express my gratitude to Natalia Díaz-Rodríguez and Giovanni Maria Farinella for having taken the time to accurately read this thesis. It was a great honor for me to receive their valuable feedback to improve the manuscript.

My path has been amazing thanks to the incredible fellows along the way. I really enjoyed being part of the LTTM Lab. I would like to thank Gianluca Agresti and Sebastiano Verde for having warmly welcomed me in the lab when I first arrived and for all the unforgettable memories we shared. Thanks to Marco Toldo and Francesco Barbato for going through my attempts to become a better supervisor, to Adriano Simonetto and Fabio Capraro for working closely with me.

Thanks to all my co-authors, who enriched my knowledge and/or trusted me as co-supervisor for their projects: Gianluca Agresti, Marco Toldo, Andrea Maracani, Francesco Barbato, Elena Camuffo, Donald Shenaj, Edoardo Borsato, Luca Rossi, Giampaolo Pagnutti, Matteo Biasetton, Mazen Mel, Giulia Rizzoli, Teo Spadotto, Maria Camporese, Andrea Agiollo. Thank you, it was a great pleasure to work with you.

During these years, I had the chance to meet wonderful colleagues and friends, which contribute to such an amazing journey: Francesco, Mattia, Alberto, Matteo, Paolo, Martina, Davide, Federico, Tommaso, Matteo, Luca, Silvia, Leonardo, Marco, Daniel, Chiara, Michele, Giulia, Elvina, Alessandro. For the frequent insightful discussions on deep learning and wonderful moments around the world: thanks to Aga, Fabio, Ettore and Guglielmo.

On the personal side, thanks to all my long-time friends who always support me from around the world: Nicola, Lucrezia, Marco, Sophia, Enrico. Thanks to my tennis buddy Giovanni and to my ping-pong buddy Federico. Every time we can reunite it is always a great feeling as if we have never been far apart. Thank you.

Last but not least, I would like to thank my family, from my cousins to my grandparents, for always believing in me. A deep gratitude goes to my parents, Fabio and Elisabetta, for letting me freely pursue this path and always supporting me. Grazie!