This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3345152

Learning from Mistakes: Self-Regularizing Hierarchical Representations in Point Cloud Semantic Segmentation

Elena Camuffo, *Student Member, IEEE*, Umberto Michieli, Simone Milani, *Member, IEEE*

***Abstract*—Recent advances in autonomous robotic technologies have highlighted the growing need for precise environmental analysis. Point cloud semantic segmentation has gained attention to accomplish fine-grained scene understanding by acting directly on raw content provided by sensors. Recent solutions showed how different learning techniques can be used to improve the performance of the model, without any architectural or dataset change. Following this trend, we present a coarse-to-fine setup that LEArns from classification mistaKes (LEAK) derived from a standard model. First, classes are clustered into macro groups according to mutual prediction errors; then, the learning process is regularized by: (1) aligning class-conditional prototypical feature representation for both fine and coarse classes, (2) weighting instances with a per-class fairness index. Our LEAK approach is very general and can be seamlessly applied on top of any segmentation architecture; indeed, experimental results showed that it enables state-of-the-art performances on different architectures, datasets and tasks, while ensuring more balanced class-wise results and faster convergence.**

***Index Terms*—Point Clouds, Semantic Segmentation, Representation Learning, Spectral Clustering, Prototypes, Fairness.**

I. Introduction

SEMANTIC scene understanding is a challenging computer vision problem that finds application in various fields including autonomous driving, robot sensing, and virtual reality.

Specifically, semantic segmentation is the most fine-grained scene understanding task that provides point-wise labeling on image pixels or 3D points. Since the refinement of classification accuracy can bring immeasurable benefits on different tasks (from navigation control to action planning), recent research has focused on improving different deep learning models for heterogeneous types of sensed data (*e.g.*, RGB images, depth maps, and point clouds). Such improvements can be achieved in different ways. One possibility involves the adoption of enhanced processing architectures [38] that better parameterize the input data with respect to the segmentation task. In this case, it is possible to improve the results sensibly at the cost of higher computational loads and memory requirements. Current state-of-the-art solutions are typically built on the top of autoencoder architectures or fully-convolutional models [38], and their inner structure strongly depends on the task and the properties of the processed data, without relying on specific class-conditional constraints or abstractions.

All the authors are with the University of Padova.
Elena Camuffo is the corresponding author.
E-mail: {elena.camuffo, umberto.michieli, simone.milani}@dei.unipd.it

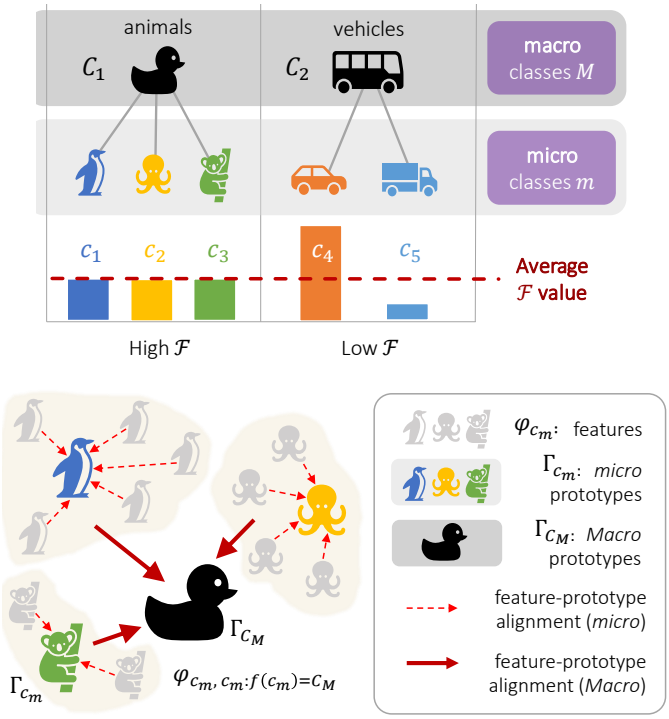


Fig. 1: We identify semantic *macro* communities (*e.g.*, *vehicles*) of *micro* classes (*e.g.*, *car* and *truck*) automatically analyzing the accuracy results of any semantic segmentation model. We regularize model training with 2 components. Top: a macro-aware fairness ($\mathcal{F}$) score on the micro classes promotes homogeneous scores within each macro cluster. Bottom: class-conditional latent features-to-prototype alignment at 2 levels (micro and macro) improves class-wise features discrimination.

A possible alternative consists in designing more accurate learning paradigms that are able to self-optimize the final performance even if architecture and datasets remain the same (without any additional intervention or information by the designer). To this extent, recent works [23], [34], [82] have focused on the adopted learning paradigms that maximize the semantic segmentation performance without increasing the size or complexity of the network. Such approaches exploit either body-edge features [34], self-supervised depth estimation [23], or pseudo labels [82], and do not perform any adaptive self-regularization estimated from a preliminary class-conditional accuracy. The main advantage of learning-based enhancement is that the training process structure the data in a self-supervised manner, in such a way that the memory and computational requirements remain the same at inference time, but the final

accuracy improves. This proves to be extremely desirable in many application scenarios (*e.g.*, embedded deep learning scheme with real-time constraints) where low latency and a limited network size can not be bargained with improved performance.

Following this trend, we propose LEAK (LEArning from mistaKes), a novel coarse-to-fine learning strategy that automatically optimizes the performance of a semantic segmentation network by shaping class subspaces according to classification errors and unbalancing. Remarkably, our approach does not imply additional supervision by human operators, making the full training process self-organized. The core idea relies on dividing the feature space in micro and *macro* regions (where *macro* space is an aggregation of *micro* class subspaces fused according to their mutual misclassification probability) and balance them according to their representativeness.

First, we use a pre-trained standard segmentation model to derive a confusion matrix over the (*micro*) classes contained in our dataset. Then, the optimization routine identifies *macro* classes that include visually similar *micro* classes by means of spectral clustering [1] on the confusion matrix. Empirically, we verified that *macro* classes include similar semantic content, as expected.

Such *micro-macro* partitioning is used to derive different regularization terms. A fairness-enforcing constraint is included in the training loss to make classification errors uniformly distributed regardless of the sample frequency or accuracy per class (upper part of Fig. 1), and hierarchy-aware class-conditional regularization constraints are introduced to embed feature vectors of the same class tightly around their micro and macro prototypical representations (lower part of Fig. 1).

We tested LEAK on different point cloud semantic segmentation networks (RandLA-Net [24], Cylinder3D [81], RangeNet++ [43]) and datasets, including sequential LiDAR data (SemanticKITTI [4]) and static datasets (Semantic3D [20], S3DIS [3]) acquired with laser scanners and other technologies. The proposed framework can be seamlessly adapted to different scenarios and is agnostic to the architecture and dataset. Its generality was also verified by adapting it to a standard semantic image segmentation dataset, *i.e.*, Pascal-VOC2012 [16], using DeepLabV3 [10].

Some recent approaches [42], [47], [63], [80] have highlighted the opportunities of a feature-level clustering using class prototypes to characterize a generic feature for each class; however, experimental results show that hierarchy-awareness can significantly boost semantic recognition.

We can summarize the main contributions of this work as follows. (1) We propose a general framework for semantic segmentation, adaptable to different experimental scenarios. In this way, LEAK proves to be very general and adaptable to different contexts. (2) We identify a semantically-consistent partition of classes in *macro* categories by inspecting the confusion scores of a standard model via spectral clustering. Previous coarse-to-fine approaches [40], [56], [57] usually need human supervision in defining the hierarchical split. LEAK adopts a fully automated procedure to extract and concretely encode such information, constraining the network with output-level fairness and feature-level regularization. (3) We devise a hierarchy-aware fairness constraint to balance classification scores regardless of the frequency or accuracy of each class. Output-level fairness has been widely investigated in resource allocation [27]; however, no prior works include fairness measures in loss definitions for training deep architectures. (4) We compute a class-conditional hierarchical prototype structure that enforces an alignment of the generated feature vectors around their prototypical representation. Some recent approaches [42], [47], [63], [80] have highlighted the opportunities of a feature-level clustering to characterize a generic feature for each class. However, experimental results show that hierarchy awareness can significantly boost semantic recognition. (5) We benchmark our approach on different standard point cloud and RGB semantic segmentation datasets outperforming state-of-the-art architectures.

II. Related Work

Point Cloud Semantic Segmentation (PCSS) has been tackled using different methods and architectures [8]. A first set of approaches relies on discretization methods that transform point clouds into discrete data structures. These structures can be dense, like voxels [60] or octrees [51] or sparse, like permutohedral lattices [52] and can be treated as three-dimensional images where convolutions can be applied. Another category of methods consists in projecting the point cloud on a bi-dimensional structure to infer predictions and map it back in a later stage. The projection methods are based either on multi-view [58], spherical [43] or cylindrical [74] projections. Deep learning architectures used in these cases are usually well-established convolutional neural networks (CNN) pre-trained on image datasets. Compared with discretization-based models, these methods are able to improve the performance for different tasks by taking multiple views of the object or scene of interest. In addition they are efficient in terms of computational complexity. Finally, point-based methods avoid limitations posed by both projection- and discretization-based methods, *e.g.*, loss of structural information, via direct processing of the raw point cloud data. Among these methods we can distinguish point-wise MLP approaches [24], [48], [49], point-convolutions [35], RNN-based [26], [71] and graph-based methods [32], [67].

However, the most recent approaches rely on 4D convolutions [17] or transformers [19], [31] to accomplish the task and they need a huge computational power and storage capacity. More lightweight approaches consist in a mixture of methods; for example, some architectures provide voxel-wise predictions refined with point-wise labels [81]. These approaches have been recently exploited with Knowledge Distillation methods that combine the two data representations in order to improve performance [22].

Prototype-based regularization strategies have been successfully proposed to support deep learning architectures in solving a wide range of transfer learning problems, such as few-shot learning [2], [9], [11], [15], [55], [70], domain adaptation [45], continual learning [13], federated learning [41]. The common idea is to regularize model training by

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3345152

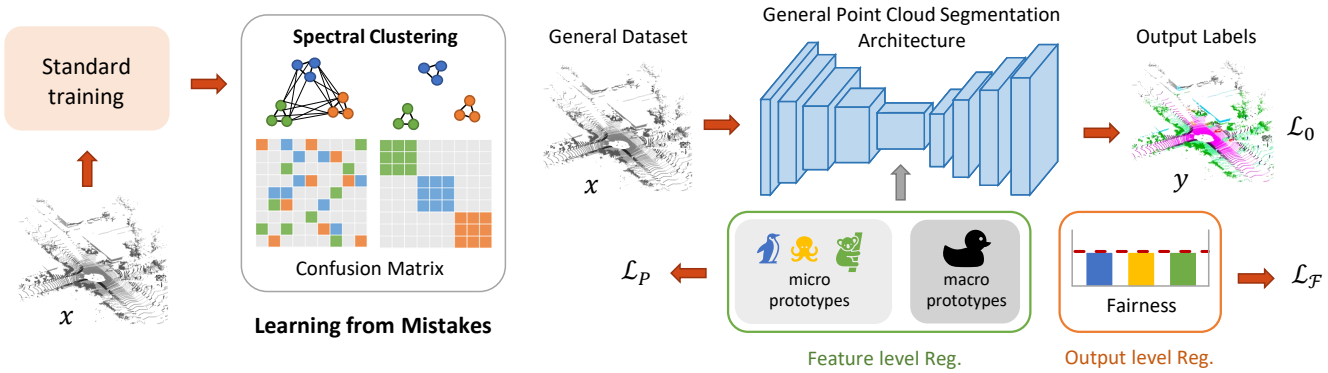


Fig. 2: Overall pipeline of the proposed approach. First (left side), we analyze the results of a standard supervised learning performed by any off-the-shelf segmentation model identifying *macro* communities of similar *micro* semantic classes. Then (right side), we regularize the learning of the model by clustering features around their prototypical semantic representation at two levels (micro and macro) and by a macro-aware fairness score on the micro classes.

constraining the space of the extracted features. Such techniques found applications in image classification [29], [33], [45], [55], image semantic segmentation [15], [80] and 3D point cloud segmentation [11], [79].

In [55] the idea that there exists an embedding in which points cluster around a single prototype representation for each class is first formulated, and then features are assigned to the class of the closest prototype (nearest neighbor). Since then, prototypes have been employed in many ways. Some works employ prototypical contrastive learning by clustering features of the same class tightly around their prototype, while spacing apart features of different classes [33]. Matching of prototypes improved generalization across domains [36], [45] and reduced forgetting when distilling knowledge from a support set of prototypes [9], [41].

Multiple class-conditional prototypical representations have been employed in [2], [70], [79] to better capture the complex statistical distribution of the extracted features. However, to the best of our knowledge, no prior work investigates the interaction of coarse- and fine-level prototypes to leverage standard supervised model training. For point clouds, prototypes have been used in [33] to support few-shot PCSS by either composing fine-level prototypes [33] or by building an attention mechanism from multiple prototypes [79].

Hierarchical composition of semantic representations has been explored in few previous work for part-based regularization [33], [70] and coarse-to-fine approaches where coarse-level classes are refined into finer categories [40], [56]. However, they all require an explicit *micro-to-macro* assignment.

III. Methodology

Given an input point cloud, *i.e.*, a set of N 3D points $\mathrm{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$, and a set of candidate semantic labels $\mathrm{Y} = \{y_1, y_2, \ldots, y_N\}$, the objective of semantic segmentation is to associate each input point $\mathbf{x_i}$ with a semantic label y_i.

Such input points can be treated in several ways: (1) they can be discretized as voxels, (2) they can be projected and treated as 2D images, (3) or they can be processed directly as they are. We devise experiments on each of the three methodologies, addressing the main focus on the direct processing method. In the following paragraphs we provide a detailed explanation of each component, *i.e.*, clustering on the standard model mistakes, class balancing through fairness enforcement, and feature-prototype alignment in the latent space. Note that the first two components are totally independent of the model, while the latter weakly depends on how the construction of prototypical features occurs within the architecture, which in turn depends on the processing method used for point clouds.

An overall scheme of our approach is shown in Fig. 2. Spectral clustering is applied on the confusion matrix inferred from the standard pre-trained model. The hierarchical partition obtained over the set of classes is used within the fairness objective at the output and to build prototypes at both *micro* and *macro* levels. This way the model *learns from mistakes* by adopting a semantic-driven self-regularization approach, and obtains an overall improvement against the standard solution.

A. Learning from Mistakes

The first building block of our LEAK is the effective core of the self-regularization strategy based on mutual semantic misclassifications that *learns from mistakes*. Generally, standard supervised approaches train models from scratch relying on an annotated training set; coarse-to-fine hierarchical approaches exploit additional information, grouping ground truth labels *a priori* into several *macro* categories [56], generated via a human annotation activity. Conversely, LEAK performs *a posteriori* unsupervised clustering of classes, independently from the specific dataset and architecture. Indeed, the class partition is derived from the misclassifications produced by the standard segmentation method. Such errors provide meaningful feedback about the feature space organization and allow highlighting of the classes that can be easily confused. Therefore it permits an optimization of their hidden representations enhancing the separation between the corresponding semantic regions.

A pre-trained standard segmentation model is employed to infer predictions on the validation set, computing the confusion matrix over classes. This matrix $\mathcal{A}$ is considered as an adjacency matrix associated with a complete graph network $\mathcal{G}$, where the different classes are assigned to nodes and the conditional error probabilities are the edge weights. We identify the nodes of $\mathcal{G}$ with $\{c_i\}, i \in [0, m)$, where m is the total number of classes, and the edges with $\{d_{i,j}\}, i, j \in [0, m)$, where i, j denote the ground truth and predicted class index, respectively. The edge

$d_{i,j}$ is associated with the probability of classifying ground truth class c_i into predicted class c_j. We use this representation to draw the subdivision in communities with a clustering algorithm, identifying M clusters, defined by $\{C_i\}, i \in [0, M)$.

Specifically, this partitioning is performed via *spectral clustering* commonly used to identify communities of nodes in a graph based on the edges connecting them. The adjacency matrix $\mathcal{A}$ is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset. The algorithm follows an iterative procedure that exploits the eigenvalues of the similarity matrix to conduct dimensionality reduction and progressively subdivides the network into two clusters until the optimal number of communities is reached. This number is estimated *a priori* thanks to the graph conductance measure [28] where the optimal number of clusters corresponds to the number of local minima. The communities found at this step represent the macro-grouping of classes. The left side of Fig. 2 shows a visual representation of the effects brought by the spectral clustering algorithm on the graph and the confusion matrix. Each color corresponds to a set of nodes (*i.e.*, *micro* classes) that belong to the same cluster (*i.e.*, *macro* class). The tree structure of the left side in Fig. 1 is derived bottom-up with this approach and shows the classes' hierarchical organization.

B. Feature-Prototype Alignment

Prototypes (*i.e.*, class centroids) are non-learnable vectors in the feature space that are representative of each semantic category that appears in the dataset [44], [72], [80]. During training, the features extracted by the encoder contribute in forming the latent prototypical representation both for micro and macro classes. Class prototypes $\mathbf{\Gamma}_c$ ideally represent the features' objective for the respective class at each training step.

Their computation occurs in place with a running average updated at each training step with supervision. At training step t with batch $\mathcal{B}$ of B total samples, the prototypes are updated for a generic class c as:

$$\mathbf{\Gamma}_c[t] = \frac{1}{k_c[t]} \left(k_c[t-1] \cdot \mathbf{\Gamma}_c[t-1] + n_c \sum_{\hat{\boldsymbol{\varphi}}_c \in \mathcal{B}} \hat{\boldsymbol{\varphi}}_c \right) \quad (1)$$

where $\hat{\boldsymbol{\varphi}}_c$ is a feature vector in the current batch $\mathcal{B}$ corresponding to class c, $k_c[t]$ is the number of feature vectors corresponding to class c met in all previous batches, and n_c is the number of feature vectors corresponding to class c in current batch $\mathcal{B}$. Therefore, $k_c[t] = k_c[t-1] + n_c$ with $k_c[0] = 0$.

The correspondence of each feature vector $\hat{\boldsymbol{\varphi}}$ with class c is based on the idea that a general encoder network preserves local structures from the input space, whether it is composed of convolutional or MLPs' layers. Therefore, the ground truth labels of each point cloud are tracked throughout the encoder to reach the latent space and provide semantic labels for it. Then, features with the same semantic class c are aggregated to contribute in the construction of prototype $\mathbf{\Gamma}_c$.

Class prototypes are initialized to $\mathbf{\Gamma}_c[0] = 0 \ \forall c \in [0, m)$.

We use the l_1 norm $||\cdot||_1$ as metric distance. We report in *Suppl. Mat.* an ablation on the loss function used. The specific loss function is defined as:

$$\mathcal{L}_{P_m} = \frac{1}{m} \sum_{c=0}^{m-1} \frac{1}{n_c} \sum_{\hat{\boldsymbol{\varphi}}_c \in \mathcal{B}} ||\mathbf{\Gamma}_c - \hat{\boldsymbol{\varphi}}_c||_1. \quad (2)$$

The integration of prototypical representations in the objective function promotes a self-driven progressive regularization of the latent space, forcing an alignment of new incoming features to their prototypes (lower of Fig. 1). We compute prototypes both at micro and macro levels.

Macro prototypes are obtained following Eq. (1), but considering the *macro* class $C = f(c)$ instead of the *micro* class c, with $f(\cdot)$ being the *micro-to-macro* mapping function identified by our spectral clustering algorithm. The respective loss function $\mathcal{L}_{P_M}$ is drawn as in Eq. (2), considering M *macro* classes indexed by C.

Including both micro and macro prototypes, we increase simultaneously coarse and fine expressiveness (division in well-separated clusters) for the latent representations, promoting a meaningful hierarchical organization of feature samples. The addition of feature-level regularization has shown to be beneficial also for different tasks such as image semantic segmentation [69].

Note that features are latent representations of input points, but they have usually lower resolution and have different shapes with respect to the input tensors. To assign feature labels we must account for the specific encoder structure and sub-sampling method. Therefore, feature labels are evaluated by propagating input labels through the encoder. The three selected architectures represent exemplar networks of the three most common methods to process point clouds: RandLA-Net [24] is a point-based method built of MLP layers and sub-samples points using random sampling; Cylinder3D [81] partitions the 3D space discretizing point clouds with cylindrical voxel-like structures; RangeNet++ [43] projects point clouds on 2D surfaces to process them like images.

C. Attentive Fair Weighting

To enforce the regularization effect of feature-prototype alignment, an attentive per-class weighting scheme is introduced. This constraint is derived from the experimental observation that the number of points per class plays a significant influence on the classification accuracy for that class. For example, in SemanticKITTI [4] the most frequent classes, *e.g.*, *vegetation* or *road*, obtain higher accuracy with respect to the least frequent ones, *e.g.*, *person*, independently on the underlying architecture. Besides, in many practical applications, the least frequent classes are the most critical ones (*e.g.*, *person* in an automotive scenario).

We propose a regularization objective derived from the Jain's fairness index $\mathcal{F}$ [27] to provide a balanced per-class weighting. In other words, we address a resource allocation problem within each *macro* class, considering *micro* classes belonging

TABLE I: Per-class IoU on SemanticKITTI [4] dataset. †: model re-trained from the official codebase for a fair comparison. **Bold** indicates best compared to baseline.

	mIoU	road	swalk	parking	ot-ground	building	car	bike	mbike	truck	ot-vehicle	vegetation	trunk	terrain	person	bicyclist	mcyclist	fence	pole	t-sign
PointNet [48]	14.6	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPG [32]	17.4	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
PointNet++ [49]	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
TangentConv [59]	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
DarkNet21Seg [4]	47.4	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0
DarkNet53Seg [4]	49.9	91.8	74.6	64.8	27.9	84.1	86.4	25.5	24.5	32.7	22.6	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	52.2
RangeNet53++ [43]	52.2	91.8	75.2	65.0	27.8	87.4	91.4	25.7	25.7	34.4	23.0	80.5	55.1	64.6	38.3	38.8	4.8	58.6	47.9	55.9
LatticeNet [52]	52.9	90.0	74.1	59.4	22.0	88.2	92.9	26.6	16.6	22.2	21.4	81.7	63.6	63.1	35.6	43.0	46.0	58.8	51.9	48.4
ThickSeg [18]	55.2	71.2	27.2	27.0	70.5	55.1	45.1	27.3	48.7	79.4	66.6	73.1	63.6	74.5	58.9	69.1	35.9	73.4	32.3	50.9
FPS-Net [68]	57.1	91.1	74.6	61.9	26.0	87.4	91.1	37.1	48.6	37.8	30.0	80.9	61.2	65.0	60.5	57.8	7.5	57.4	49.9	59.2
BAAF-Net [50]	59.9	90.9	74.4	62.2	23.6	89.8	95.4	48.7	31.8	35.5	46.7	82.7	63.4	67.9	49.5	55.7	53.0	60.8	53.7	52.0
M-RangeSeg [66]	61.0	93.9	50.1	43.8	43.9	43.2	63.7	53.1	18.7	90.6	64.3	74.6	29.2	91.1	64.7	82.6	65.5	65.5	56.3	64.2
RangeNet21† [43]	46.1	**93.7**	81.0	43.3	**0.0**	**83.3**	**93.7**	**23.4**	**38.8**	**19.9**	12.2	**81.8**	46.0	**72.4**	27.3	**49.9**	**0.0**	48.8	36.6	28.3
LEAK (RangeNet21†)	**46.4**	93.6	**81.2**	**46.9**	**0.0**	82.8	90.8	22.0	37.8	8.7	**16.6**	81.5	**46.7**	**72.4**	**28.8**	49.7	**0.0**	**52.5**	**39.5**	**29.4**
RandLA-Net† [24]	53.4	**91.8**	77.0	**41.0**	1.0	**87.7**	**93.4**	13.4	30.3	69.9	39.5	84.9	**60.4**	73.4	50.6	67.2	**0.0**	43.7	50.9	37.9
LEAK (RandLA-Net†)	**54.7**	91.4	**77.2**	39.9	**1.6**	**87.7**	93.3	**17.5**	**32.4**	**78.0**	**42.0**	**85.3**	58.0	**73.9**	**54.2**	**69.6**	**0.0**	**43.9**	**52.8**	**40.5**
Cylinder3D† [81]	64.7	**94.4**	81.1	49.0	0.3	**89.5**	**96.9**	39.1	64.3	**87.3**	**63.9**	87.4	69.0	72.7	74.8	90.6	**0.1**	**55.6**	**64.4**	49.2
LEAK (Cylinder3D†)	**65.2**	94.2	**84.3**	**49.2**	**2.6**	86.2	96.5	**48.7**	**65.3**	85.8	60.6	**88.0**	**69.2**	**85.1**	**75.7**	**91.6**	0.0	38.7	65.3	**51.8**

to the same *macro* class as the users that are sharing the same resource:

$$\mathcal{F} = \sum_{C=0}^{M-1} \frac{\left(\sum_{\boldsymbol{\pi}_c \in \mathcal{B}, c:f(c)=C} \pi_{c,c}\right)^2}{m_C \cdot \sum_{\boldsymbol{\pi}_c \in \mathcal{B}, c:f(c)=C} \pi_{c,c}^2} \tag{3}$$

where m_C is the number of (*micro*) classes within *macro* class C, $\pi_{c,c}$ represents the c-th element in vector $\boldsymbol{\pi}_c$, and $\boldsymbol{\pi}_c$ is the average prediction vector for class c, obtained as:

$$\boldsymbol{\pi}_c = \frac{1}{\bar{n}_c} \sum_{\boldsymbol{p}_c \in \mathcal{B}} \boldsymbol{p}_c \tag{4}$$

where $\boldsymbol{p}_c$ is a generic prediction vector with ground truth class c and $\bar{n}_c$ is the number of points labeled as c in the current batch $\mathcal{B}$.

A high fairness value denotes a truly balanced resource allocation among the entities, while low values of fairness show an unbalanced share of resources (upper part of Fig. 1).

Therefore, in order to preserve accuracy homogeneity among classes, we design a fairness-based loss function as follows:

$$\mathcal{L}_{\mathcal{F}} = (1 - \mathcal{F}). \tag{5}$$

While prototype alignment forces a semantic target representation for each class at *micro* and *macro* levels, the attentive weighting constraint based on fairness aims at providing unbiased output predictions within each *macro* class. In other words, by posing such weighting constraint to predictions of the same *macro* class, we give the same importance to all the semantically consistent *micro* classes.

D. Objective Function

The training objective is given by the combination of the base loss function for each architecture ($\mathcal{L}_0$) with the additional objective given by the LEAK components. The base loss function depends on the selected architecture. It corresponds to the standard cross-entropy loss with inverse class weighting for RandLA-Net [24] and RangeNet++ [43], to the Lovasz-softmax loss [5] for voxel features plus the cross-entropy loss with inverse class weighting for point-feature refinement in Cylinder3D [81], and to the plain cross-entropy loss for DeepLabV3 [10].

The LEAK components are given by micro-level ($\mathcal{L}_{P_m}$) and macro-level ($\mathcal{L}_{P_M}$) feature-prototype alignment objectives, and a class-wise attentive weighting constraint ($\mathcal{L}_{\mathcal{F}}$). The full objective is then computed as:

$$\mathcal{L}_{LEAK} = \mathcal{L}_0 + \lambda_{P_m} \cdot \mathcal{L}_{P_m} + \lambda_{P_M} \cdot \mathcal{L}_{P_M} + \lambda_{\mathcal{F}} \cdot \mathcal{L}_{\mathcal{F}} \tag{6}$$

where the balancing hyper-parameters have been tuned using a validation set.

IV. Training Procedure

We experiment on publicly available benchmarks, using three point cloud datasets (2 outdoor and 1 indoor) and one image dataset.

The **SemanticKITTI** [4] dataset consists of 43552 densely annotated outdoor LiDAR scans. The training split contains 19130 scans, while the validation split 4071 scans (that we used for testing, as done by all competing works being the test labels not publicly available). The mean Intersection over Union (mIoU) score over 19 categories is used as the standard metric and results are reported on the original validation set.

The **Semantic3D** [20] dataset consists of outdoor static point clouds, 15 for training and 15 for online testing. Each point cloud has up to 108 points. We only use color and spatial coordinates to train and test our models, following previous work [24]. We evaluate the performance via mIoU and OA on the 8 classes.

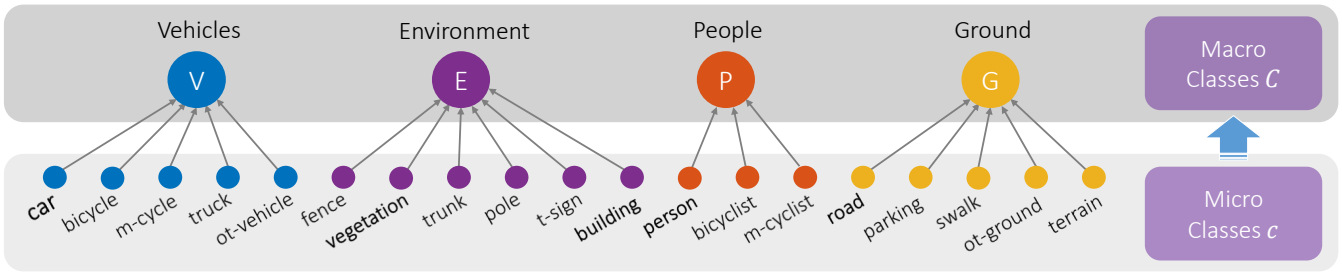


Fig. 3: Hierarchical *a posteriori* organization of SemanticKITTI [4] classes.

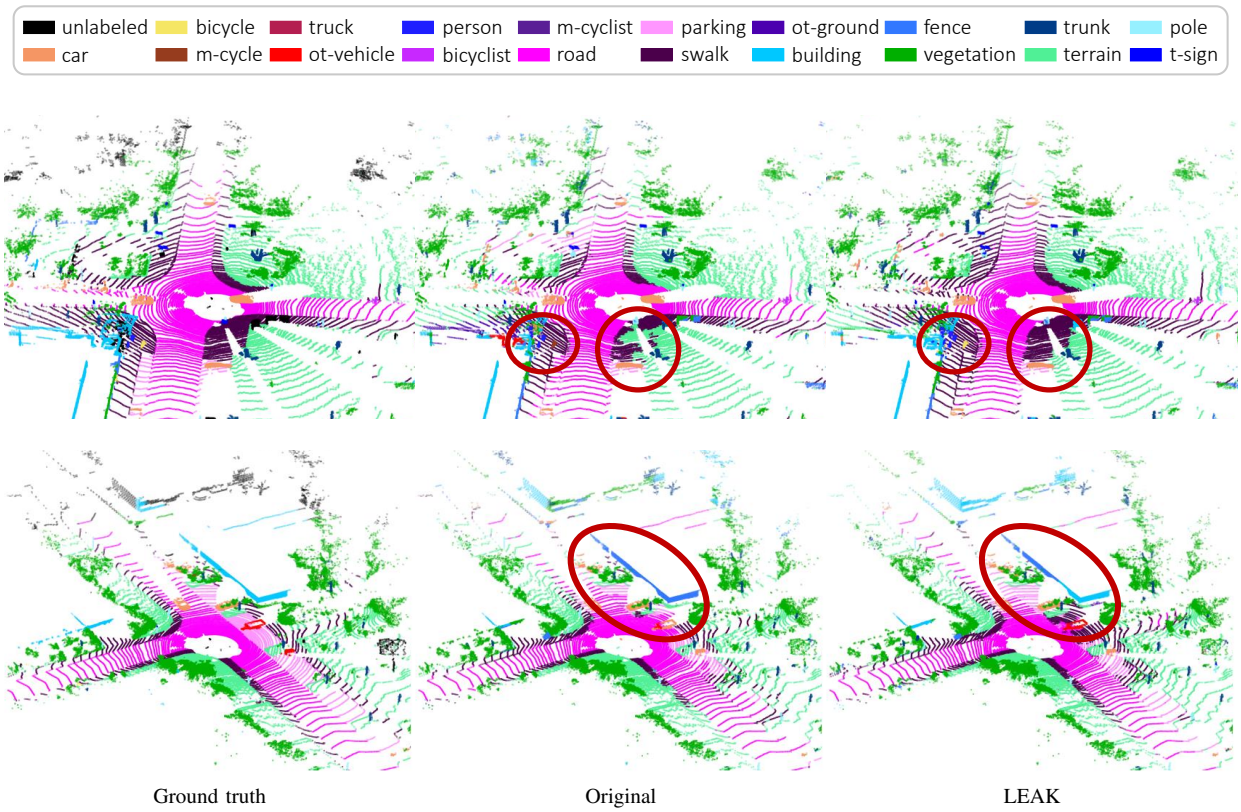


Fig. 4: Qualitative results from SemanticKITTI [4] with RandLA-Net [24].

The **S3DIS** [3] dataset consists of 271 indoor scans of medium-sized single rooms, with dense 3D points. We use the standard 6-fold cross-validation in our evaluation. We report mIoU, mean class Accuracy (mAcc), and Overall Accuracy (OA) of the 13 classes.

The **PascalVOC2012** [16] is used to validate our methods on RGB image samples. It contains 10582 images for training and 1449 for online testing. We use the mIoU to evaluate the performance on the 21 different classes.

Our proposed strategy is agnostic to the backbone architecture. To prove it, in our experimental evaluation we use **RandLA-Net** [24], **RangeNet++** [43], **Cylinder3D** [81] and **DeeplabV3** [10] with their original hyper-parameters configuration. We train RandLA-Net optimizing the network weights following [24], with Adam optimizer and the same learning rate policy, momentum, and weight decay. The initial learning rate is set to 10^{-2}, and decreased with a polynomial decay rule with power 0.95. In each learning step, we train the models for 100 epochs with a batch size of 6 for SemanticKITTI and S3DIS, and 4 for Semantic3D, as in the original model. For Cylinder3D we train the model on SemanticKITTI using the standard configuration for 40 epochs with batch size 2 and learning rate 10^{-3}. Finally, for the evaluation on DeepLabV3 we follow the original configuration [10] and we trained the model for 30 epochs with initial learning rate 7×10^{-4} and batch size 6. We use an NVIDIA GeForce RTX 3090 GPU with CUDA 10.2 to train all the models.

V. Experimental Results

We extensively evaluate the performance of our approach with different architectures and datasets, analyzing each component of LEAK separately.

First of all, we show the results on SemanticKITTI [4] dataset with two architectures: namely, RandLA-Net [24] and Cylinder3D [81]. Then, we tested LEAK on two different datasets: S3DIS [3] and Semantic3D [20].

TABLE II: Quantitative results on the Semantic3D [20] dataset with RandLA-Net [24]. †: model re-trained from the official codebase for a fair comparison. **Bold** indicates best compared to baseline.

	OA	mIoU	mm-terrain	nat-terrain	h-vegetation	l-vegetation	buildings	hard scape	s-artefacts	cars
SnapNet [6]	88.6	59.1	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
SEGCloud [60]	88.1	61.3	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
RF MSSF [61]	90.3	62.7	87.6	80.3	81.8	36.4	92.2	24.1	42.6	56.6
MSDeepVoxNet [53]	88.4	65.3	83.0	67.2	83.8	36.7	92.4	31.3	50.0	78.2
ShellNet [75]	93.2	69.3	96.3	90.4	83.9	41.0	94.2	34.7	43.9	70.2
GACNet [65]	91.9	70.8	86.4	77.7	88.5	60.6	94.2	37.3	43.5	77.8
SPG [32]	94.0	73.2	97.4	92.6	87.9	44.0	83.2	31.0	63.5	76.2
KPConv [62]	92.9	74.6	90.9	82.2	84.2	47.9	94.9	40.0	77.3	79.7
BAAF-Net [50]	94.9	75.4	97.9	95.0	70.6	63.1	94.2	41.6	50.2	90.3
RandLA-Net† [24]	90.8	75.1	93.4	75.6	90.0	**58.7**	**87.7**	32.0	**83.0**	80.3
LEAK (RandLA-Net†)	**90.9**	**76.1**	**93.6**	**75.9**	**90.4**	57.2	88.5	**35.5**	80.8	**87.1**

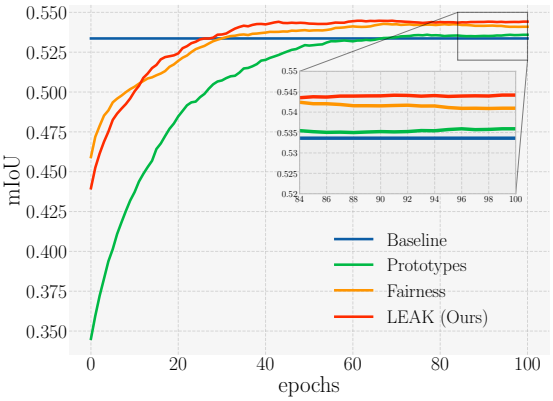


Fig. 5: mIoU curves comparing reference value (blue) to supervised training with the addition of prototype regularization (green), fairness (orange), or both (red). Curves smoothed via running average filter with window size 12. LEAK provides higher mIoU and at the same time faster convergence speed.

TABLE III: Quantitative results on the S3DIS [3] dataset (6-fold cross validation) with RandLA-Net [24]. †: model re-trained from the official codebase for a fair comparison.

	OA	mAcc	mIoU
PointNet [48]	78.6	66.2	47.6
PointNet++ [49]	81.0	67.1	54.5
DGCNN [67]	84.1	—	56.1
SPG [32]	85.5	73.0	62.1
DSPoint [73]	—	70.9	63.3
PointCNN [35]	88.1	75.6	65.4
PointWeb [76]	87.3	76.2	66.7
ShellNet [75]	87.1	—	66.8
KPConv [62]	—	79.1	70.6
JSNet [78]	71.7	88.7	61.7
FKAConv [7]	—	—	68.4
Point-PlaneNet [46]	—	92.1	54.8
JSENet [25]	—	—	67.7
CT2 [39]	—	—	67.4
RandLA-Net† [24]	87.6	84.8	68.0
LEAK (RandLA-Net†)	**88.1**	**85.4**	**68.5**

The robustness of the method is further studied by applying LEAK in a different domain: we perform image semantic segmentation using DeepLabV3 [10] on PascalVOC 2012 [16] dataset. Finally, the effect of each component is analyzed separately in an extensive ablation study.

SemanticKITTI. We start our analyses on the public SemanticKITTI [4] benchmark, reporting the results in Tab. I, where our approach is compared against several well-established architectures. We took two among the most successful architectures and we applied LEAK during their model training. The introduction of LEAK components improved the mIoU by 1.3% on RandLA-Net [24] and by 0.7% on Cylinder3D [81]. The performance drop in the retrained baselines are due to the use of a more recent version of CUDA library and other packages. More information about specific packages and versions are provided in the code repository.

However, training with LEAK shows considerable improvements that outperform the baseline solution for both RandLA-Net and Cylinder3D. Note that in RandLA-Net the prototypical representations are built based on the point-wise features, while in Cylinder3D on the voxel-wise ones.

Looking at the per-class scores, it emerges how LEAK can provide more balanced results across the classes, thanks in particular to the fairness constraint. The automatic hierarchical grouping for SemanticKITTI classes is shown in Fig. 3, where we can appreciate that each *macro* class contains semantically similar *micro* classes. It is also possible to notice that the per-class IoU increases on the least populated classes (made exception for *truck* in RangeNet21). In this case, fairness weighting combined to a higher misclassification probability depending on misleading object shapes after projection (RangeNet is based on a set of convolutional layers applied on images after spherical projections) induces a higher error probability on the points along the border of the object. This inconvenience is solved whenever applying LEAK to the other two architectures (RandLA-Net and Cylinder3D) that process array points in 3D coordinates.

Fig. 4 shows some qualitative results that compare the predictions of RandLA-Net trained with or without our LEAK. The most relevant improvements are highlighted with red circles. We observe that prediction errors of the original model often involve semantically and geometrically similar classes: *sidewalk* is misclassified as *road* in the first row, *building* is misclassified as *fence* in the second row. Our approach, instead, can correctly label these objects, thanks to the increased latent space self-regularization and fairness of accuracy across the same macro class. Indeed, *road* and *sidewalk* belong to the same *a posteriori* macro class, similarly to *fence* and *building*. Further qualitative results are provided in *Suppl. Mat.*

Moreover, to verify the robustness of LEAK, we perform additional experiments on two other datasets, with different properties and acquired with different methodologies with respect to SemanticKITTI. We select S3DIS [3] and Semantic3D [20] datasets, employing RandLA-Net. The

TABLE IV: Ablation study on SemanticKITTI [4] dataset with RandLA-Net [24]. *: measure of the prediction coherence within the macro classes.

	$\mathcal{L}_0$	$\mathcal{L}_{\mathcal{F}}$	$\mathcal{L}_{P_M}$	$\mathcal{L}_{P_m}$	SC	**mIoU**	**fwIoU**	**hIoU***
Baseline	✓					53.4	81.4	77.9
Prototypes	✓		✓	✓		53.9	81.5	**85.2**
Fairness	✓	✓				54.4	82.0	84.8
LEAK	✓	✓	✓	✓	✓	**54.7**	**82.1**	83.1

TABLE V: Recent state-of-the-art approaches results in terms of mIoU for standard method and equipped with LEAK, which is shown to boost results on every type of architecture and dataset.

Model	**Dataset**	**Std.**	**LEAK**
SphereFormer [30]	SemanticKITTI [4]	67.8	**68.5**
PVKD [22]	SemanticKITTI [4]	67.9	**68.4**
PVCNN [37]	SemanticKITTI [4]	17.2	**17.8**
Strat.Transformer [31]	S3DIS [3]	66.4	**66.9**
Pt.Transformer [77]	S3DIS [3]	70.4	**71.8**
PVCNN [37]	S3DIS [3]	55.6	**56.1**

strong generalization ability of our approach emerges clearly, despite the huge gap in the nature of the considered datasets. Indeed, the three point cloud datasets differ greatly in terms of the number of scenes, the number of points per scene, the acquisition methodologies, the environment (indoor/outdoor), and the number of classes. SemanticKITTI scenes are organized in sequences and distributed according to sparse concentric regions. Instead, static datasets present denser and regularly distributed point regions. Qualitative results are provided in *Suppl. Mat*.

S3DIS. Results on S3DIS are shown in Tab. III. RandLA-Net with LEAK achieves remarkable results and, in particular, outperforms the baseline training scheme by 0.5% of mIoU. The improvement is shown also according to the other metrics, with an increase of 0.5% and 0.8% for the OA and mAcc, respectively.

Semantic3D. The per-class results in terms of mIoU and OA achieved on Semantic3D are reported in Tab. II. RandLA-Net with LEAK outperforms all the reported approaches, boosting the baseline of 1.0% mIoU.

VI. Ablation Studies

We report an extensive ablation study to fully validate our approach. First, we show task generalizability, reporting the results on VOC2012 [16]. Then, we devise separate studies on each component of our LEAK.

A. Task generalization.

VOC2012. We show the robustness of LEAK on segmentation of RGB images rather than point clouds. To perform image semantic segmentation we consider the VOC2012 [16] dataset and we use DeepLabV3 [10], with ResNet-101 [21] as backbone (weights pre-trained on ImageNet [14]). Also in this case, LEAK brings an improvement compared to the standard method. In terms of mIoU, we obtain a value of 66.4% using the standard approach and 66.7% when LEAK is introduced.

TABLE VI: Ablation study on hierarchical grouping and weighting schemes using SemanticKITTI [4] dataset. *: coarse-to-fine is intended as a two-steps training, half of the total epochs on *macro* classes and half on *micro* classes, using LEAK class partitioning.

Model	Modifications	**mIoU**
RandLA-Net	—	53.4
LEAK (RandLA-Net)	—	**54.7**
LEAK (RandLA-Net)	*a priori* grouping	54.2
RandLA-Net	coarse-to-fine*	53.8
RandLA-Net	weight scheme of SalsaNext [12]	52.3
Cylinder3D	—	64.7
LEAK (Cylinder3D)	—	**65.2**
LEAK (Cylinder3D)	*a priori* grouping	64.2
Cylinder3D	inverse frequency weighting	64.8
Cylinder3D	weight scheme of SalsaNext [12]	63.5

Such a result further proves the validity of our method, showing that it is agnostic not only to the backbone architecture and the dataset but even, more generally, to the task.

In addition, we report in Tab. V further experiments on recent architectures for Point Cloud Semantic Segmentation showing that LEAK improves baseline performance on top of every segmentation architecture, also in case transformer-based architectures are employed.

B. LEAK Components

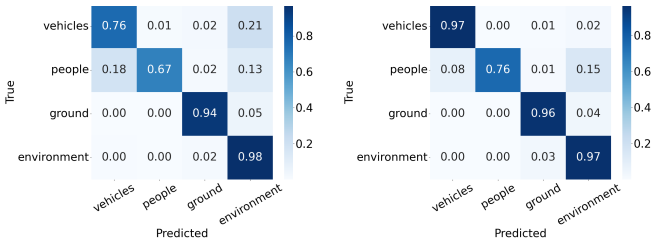
A first set of analyses were carried on the SemanticKITTI dataset with RandLA-Net architecture. Tab. IV highlights the contribution of each component of LEAK on the final model scores. We observe that both the fairness and the feature-prototypes alignment constraints significantly increase the final mIoU over the original model by 1% or 0.5% respectively. The combination of such regularization terms produces non-overlapping and mutual benefits, allowing for an increment of 1.3% in the final mIoU. Experimental results show that a fundamental requirement for the success of our approach is the *a posteriori* hierarchical clustering of the micro classes; indeed, employing a semantic-unaware random clustering of micro classes leads to a small mIoU of 51.3%, which is even lower than the original approach (-2.1%).

Also, we report two additional metrics to show the effects of each component: the frequency weighted Intersection over Union (fwIoU) that weights each class importance according to their appearance frequency, and we introduce the hierarchical Intersection over Union (hIoU), defined as:

$$\text{hIoU} = \frac{1}{M+1} \sum_{i=0}^{M} \frac{q_{ii}}{\sum_{j=0}^{M} q_{ij} + \sum_{j=0}^{M} q_{ji} - q_{ii}} \tag{7}$$

where $q = f(p)$, such that each prediction p is mapped on the respective macro class with $f(\cdot)$. This metric is computed in such a way that every predicted micro class should be equal to the target macro class that contains it. Its purpose is to underline the weight of the hierarchical self-induced assessment over the original model. In fact, the disentangling effect of fairness and prototypes lead to an improvement of 6.9% and 7.3% respectively, in terms of hIoU, which is slightly reduced in the final model ($+5.2\%$). We can appreciate improved results also

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3345152



Pre-LEAK Post-LEAK

Fig. 6: Confusion matrices of the fine-grained predictions grouped according to the *macro* classes identified *a posteriori* pre- and post- LEAK on RandLA-Net [24] with SemanticKITTI [4].

in the fwIoU ($+0.7\%$), where the main contribution is given by the class-balancing effect of fairness.

Fig. 5 compares the temporal evolution of the mIoU score over different training epochs for each ablation model. We observe that the introduction of feature-prototype alignment delays the increase in the mIoU compared to the original baseline model (green curve), but leads to an overall improvement of 0.5% mIoU against the original model in the long term. This delay is partially generated by the dominance of the new loss terms over the cross-entropy loss. In addition, this performance degradation proves initial prototypes to be quite unreliable, since they are computed just on few feature samples. As training proceeds, the prototypical features become progressively better defined and separated, leading to an increment in the overall accuracy.

Tab. VI reports further ablation results. We report tests using other cross-entropy weighting schemes (that in general helps class balancing), and tests using *a priori* standard class grouping, as in [4]. We extended the ablation on micro-macro grouping for both RandLA-Net and Cylinder3D, outperforming standard coarse-to-fine strategy and some other state-of-the-art weighting schemes, including SalsaNext [12] (*i.e.*, squared root of inverse class frequency), and inverse class frequency weighting on Cylinder3D. Our accuracy gains are robust across setups, improving training with no architectural changes.

Fig. 6 reports the confusion matrices obtained with the standard method (Pre-LEAK) and the confusion matrix analyzed at the end of the training with LEAK (Post-LEAK).

The overall error percentages are reduced and fall mostly in the *macro* categories derived with *a posteriori* clustering. In particular, we can notice that LEAK brings a great reduction in the misclassification of *vehicles* with *environment*, from 21% misclassified samples to 2%. Also, we can see an improvement in the misclassification of *people* with *vehicles*, from 18% to 8%, with the overall results on the diagonal either improving or remaining the same.

C. Feature-prototype alignment

To measure the impact of feature-prototype alignment module we compute some metrics to compare LEAK against the original model, as shown in Tab. VII. First, we define the *inter-proto angle* Θ_Γ as the angle between prototypes in the n-dimensional space, *i.e.*:

$$\Theta_\Gamma = \frac{1}{m^2} \sum_{i \neq j; i,j \in [0,m)} \frac{< \mathbf{\Gamma}_i, \mathbf{\Gamma}_j >}{||\mathbf{\Gamma}_i|| \cdot ||\mathbf{\Gamma}_j||} \tag{8}$$

in order to measure the relative distancing among prototypes. Then, we compute the *Class Center Distance* (CCD) as the average distance between features $\boldsymbol{\varphi}_c$ of class c and the average feature array $\bar{\boldsymbol{\varphi}}_c$ of class c [64] (computed at the end of training), *i.e.*:

$$\text{CCD} = \frac{1}{N} \sum_{N} \frac{1}{N_c} \sum_{c \in [0,m)} ||\boldsymbol{\varphi}_c - \bar{\boldsymbol{\varphi}}_c||. \tag{9}$$

The parameter N is the total number of samples in the dataset, while N_c is the total number of samples for class c. The purpose of CCD is to parameterize how tightly the features are clustered around their class center. Similarly, we define also the *Prototype Distance* (PD) as the average distance of the features of class c from prototype $\mathbf{\Gamma}_c$ (which is computed by a running average during the training phase), and the *Proto-Center Distance* (PCD) as the distance between $\bar{\boldsymbol{\varphi}}_c$ and $\mathbf{\Gamma}_c$. The metric values are reported in Tab. VII and confirm that feature-prototype alignment thins the regions associated to a specific class as CCD and PD are much lower when feature-prototype alignment is enabled. Additionally, the increased value of Θ_Γ in LEAK with respect to the original training highlights that a progressive spacing is introduced among prototypes thus refining the class latent feature representations.

D. Attentive fair weighting

The effect of attentive fair weighting is instead to boost the performance from the immediate beginning of the training (green curve in Fig. 5), obtaining faster convergence and overall higher precision. The attentive fair weighting curve in Fig. 5 reaches 50% mIoU in only 10 epochs, *i.e.*, the 91.4% of the final score (54.7%). We can explain this rapid increase in the mIoU with the re-weighting, computed at each training step, which forces an equalization among different classes. Fair weights computation is performed class-wise, and shows beneficial effects from the beginning, balancing some unfair weighting of the cross-entropy that privileges the most diffused classes. In the final LEAK method, where all the regularization terms are combined, the improvement due to fair weighting is slightly attenuated by feature-prototypes alignment in the first 15 epochs but performs the best in the long run, improving the mIoU of 1.3% with respect to the standard configuration. In the lower of Tab. VIII we appreciate that per-class accuracy are more balanced when fairness is enabled (see *Suppl. Mat.*). The reported metrics are computed on the per-class results of Tab. I and confirm that fair weighting balances accuracy over classes. The mean squared error (MSE) is computed with respect to the average mIoU value and minimized by the introduction of Fairness; the same observation hold for standard deviation (σ) and entropy, which is however maximized.

E. Performance

Equipping models with LEAK improves the segmentation performances of the model with only a slight additional occupancy in terms of RAM. For example, Cylinder3D [81] with

TABLE VII: Analyses on SemanticKITTI [4] and RandLA-Net [24]. Feature-prototype alignment measured by Θ_Γ, CCD, PD and PCD.

	mIoU	$\Theta_\Gamma \uparrow$	CCD $\downarrow$	PD $\downarrow$	PCD $\downarrow$
Baseline	53.4	0.78	8.04	8.61	3.84
LEAK	**54.7**	**0.89**	**1.22**	**1.15**	**0.26**

TABLE VIII: Analyses on SemanticKITTI [4] and RandLA-Net [24]. Class-balancing effect measured by standard deviation (σ), MSE and entropy.

	mIoU	$\sigma \downarrow$	MSE $\downarrow$	entropy $\uparrow$
Baseline	53.4	29.0	796.8	1.94
Fairness	**54.4**	**28.2**	**754.8**	**1.97**

SemanticKITTI [4] shows an increase in memory occupancy of $+28$ MB. This variation is almost negligible considering the size of a general point cloud segmentation network (*e.g.*, Cylinder3D architecture takes around 8 GB). Moreover, inference time is not affected by our method. Training time is slightly worse for LEAK-based methods, but the change is almost negligible. For example, Cylinder3D with SemanticKITTI takes about 1.68 seconds per iteration with LEAK, while about 1.61 seconds per iteration without LEAK.

VII. Conclusion

In this paper, we showed that accuracy, convergence time, and homogeneity can be improved for standard point cloud semantic segmentation methods, by automatically analyzing the classification mistakes of the original models. Thanks to these errors, we identify *macro* classes, hierarchically grouping sets of mutually misclassified *micro* classes. Experimental evidence showed *a posteriori* that such groups are consistent in terms of semantic characterization or classification difficulty. (*e.g.*, due to the sample frequency or sparsity). This information is exploited in model learning to regularize feature space in a two-fold manner. First, we cluster features of the same class (both macro and micro) tightly around their prototypical representations. Second, we constrained the class-wise accuracy score to be equally distributed across *micro* classes contained within the same *macro* cluster. The proposed method boosted network performance while reducing the convergence time. Also, our solution proved to be totally agnostic to the backbone architecture and dataset, and remarkably prompt to generalization, as it was adapted to 4 different datasets, 3 architectures, and 2 types of input data (*i.e.*, point clouds and RGB images).

References

[1] A. Pothen, H. Simon, K.L.: Partitioning Sparse Matrices with Eigenvectors of Graphs, vol. 11 (1990)

[2] Allen, K., Shelhamer, E., Shin, H., Tenenbaum, J.: Infinite mixture prototypes for few-shot learning. In: ICML. pp. 232–241. PMLR (2019)

[3] Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3D semantic parsing of large-scale indoor spaces. In: CVPR. pp. 1534–1543 (2016)

[4] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: ICCV. pp. 9297–9307 (2019)

[5] Berman, M., Rannen Triki, A., Blaschko, M.B.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: CVPR. pp. 4413–4421 (2018)

[6] Boulch, A., Guerry, J., Le Saux, B., Audebert, N.: Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. Computers & Graphics **71**, 189–198 (2018)

[7] Boulch, A., Puy, G., Marlet, R.: FKAConv: Feature-Kernel Alignment for Point Cloud Convolution. In: ACCV (2020)

[8] Camuffo, E., Mari, D., Milani, S.: Recent advancements in learning algorithms for point clouds: An updated overview. Sensors **22**(4), 1357 (2022)

[9] Cermelli, F., Mancini, M., Xian, Y., Akata, Z., Caputo, B.: A few guidelines for incremental few-shot segmentation. arXiv preprint arXiv:2012.01415 (2020)

[10] Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)

[11] Chen, X., Zhang, C., Lin, G., Han, J.: Compositional prototype network with multi-view comparision for few-shot point cloud semantic segmentation. arXiv preprint arXiv:2012.14255 (2020)

[12] Cortinhal, T., Tzelepis, G., Erdal Aksoy, E.: Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In: International Symposium on Visual Computing. pp. 207–222. Springer (2020)

[13] De Lange, M., Tuytelaars, T.: Continual prototype evolution: Learning online from non-stationary data streams. In: ICCV. pp. 8250–8259 (2021)

[14] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. CVPR pp. 248–255 (2009)

[15] Dong, N., Xing, E.P.: Few-shot semantic segmentation with prototype learning. In: BMVC (2018)

[16] Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)

[17] Fan, H., Yu, X., Ding, Y., Yang, Y., Kankanhalli, M.: PSTNet: Point spatio-temporal convolution on point cloud sequences. In: ICLR (2021), https://openreview.net/forum?id=O3bqkf_Puys

[18] Gao, Q., Shen, X.: Thickseg: Efficient semantic segmentation of large-scale 3d point clouds using multi-layer projection. Image and Vision Computing **108**, 104161 (2021)

[19] Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (Apr 2021). https://doi.org/10.1007/s41095-021-0229-5, http://dx.doi.org/10.1007/s41095-021-0229-5

[20] Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., Pollefeys, M.: SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. vol. IV-1-W1, pp. 91–98 (2017)

[21] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR pp. 770–778 (2016)

[22] Hou, Y., Zhu, X., Ma, Y., Loy, C.C., Li, Y.: Point-to-voxel knowledge distillation for lidar semantic segmentation. In: CVPR. pp. 8479–8488 (2022)

[23] Hoyer, L., Dai, D., Chen, Y., Koring, A., Saha, S., Van Gool, L.: Three ways to improve semantic segmentation with self-supervised depth estimation. In: CVPR. pp. 11130–11140 (2021)

[24] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In: CVPR. pp. 11108–11117 (2020)

[25] Hu, Z., Zhen, M., Bai, X., Fu, H., Tai, C.l.: Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In: ECCV. pp. 222–239 (2020)

[26] Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3d segmentation of point clouds. In: CVPR. pp. 2626–2635 (2018)

[27] Jain, R., Durresi, A., Babic, G.: Throughput fairness index: An explanation. In: ATM Forum contribution. vol. 99 (1999)

[28] Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. Journal of the ACM **51** (08 2001). https://doi.org/10.1145/990308.990313

[29] Keswani, M., Ramakrishnan, S., Reddy, N., Balasubramanian, V.N.: Proto2proto: Can you recognize the car, the way i do? In: CVPR. pp. 10233–10243 (June 2022)

[30] Lai, X., Chen, Y., Lu, F., Liu, J., Jia, J.: Spherical transformer for lidar-based 3d recognition. In: CVPR (2023)

[31] Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., Jia, J.: Stratified transformer for 3d point cloud segmentation. In: CVPR. pp. 8500–8509 (2022)

[32] Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: CVPR. pp. 4558–4567 (2018)

[33] Li, J., Zhou, P., Xiong, C., Hoi, S.C.: Prototypical contrastive learning of unsupervised representations. In: ICLR (2021)

[34] Li, X., Li, X., Zhang, L., Cheng, G., Shi, J., Lin, Z., Tan, S., Tong, Y.: Improving semantic segmentation via decoupled body and edge supervision. In: ECCV. pp. 435–452. Springer (2020)

[35] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on X-transformed points. NIPS **31**, 820–830 (2018)

[36] Liu, Y., Deng, J., Gao, X., Li, W., Duan, L.: Bapa-net: Boundary adaptation and prototype alignment for cross-domain semantic segmentation. In: ECCV. pp. 8801–8811 (2021)

[37] Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. In: NeurIPS (2019)

[38] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. pp. 3431–3440 (2015)

[39] Mazur, K., Lempitsky, V.: Cloud transformers: A universal approach to point cloud processing tasks. In: ICCV (2021)

[40] Mel, M., Michieli, U., Zanuttigh, P.: Incremental and multi-task learning strategies for coarse-to-fine semantic segmentation. Technologies **8**(1), 1 (2020)

[41] Michieli, U., Ozay, M.: Prototype guided federated learning of visual feature representations. arXiv preprint arXiv:2105.08982 (2021)

[42] Michieli, U., Zanuttigh, P.: Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In: CVPR. pp. 1114–1124 (2021)

[43] Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: IROS. pp. 4213–4220. IEEE (2019)

[44] van den Oord, A., Vinyals, O., kavukcuoglu, k.: Neural discrete representation learning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)

[45] Pan, Y., Yao, T., Li, Y., Wang, Y., Ngo, C., Mei, T.: Transferrable prototypical networks for unsupervised domain adaptation. In: CVPR. pp. 2239–2247 (2019)

[46] Peyghambarzadeh, S.M., Azizmalayeri, F., Khotanlou, H., Salarpour, A.: Point-planenet: Plane kernel based convolutional neural network for point clouds analysis. Digital Signal Processing **98**, 102633 (2020)

[47] Pinheiro, P.O.: Unsupervised domain adaptation with similarity learning. In: CVPR. pp. 8004–8013 (2018)

[48] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017)

[49] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS **30** (2017)

[50] Qiu, S., Anwar, S., Barnes, N.: Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In: CVPR. pp. 1757–1767 (2021)

[51] Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: CVPR. pp. 3577–3586 (2017)

[52] Rosu, R.A., Schütt, P., Quenzel, J., Behnke, S.: Latticenet: fast spatio-temporal point cloud segmentation using permutohedral lattices. Autonomous Robots pp. 1–16 (2021)

[53] Roynard, X., Deschaud, J.E., Goulette, F.: Classification of point cloud scenes with multiscale voxel deep network. arXiv preprint arXiv:1804.03583 (2018)

[54] Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: NIPS. pp. 4080–4090 (2017)

[55] Stretcu, O., Platanios, E.A., Mitchell, T., Póczos, B.: Coarse-to-fine curriculum learning for classification. In: ICLRW (2020)

[56] Stretcu, O., Platanios, E.A., Mitchell, T.M., Póczos, B.: Coarse-to-fine curriculum learning. ICLR Workshop on Bridging AI and Cognitive Science (BAICS) (2021)

[57] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV. pp. 945–953 (2015)

[58] Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In: CVPR. pp. 3887–3896 (2018)

[59] Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S.: Segcloud: Semantic segmentation of 3d point clouds. In: 3DV. pp. 537–547. IEEE (2017)

[60] Thomas, H., Deschaud, J.E., Marcotegui, B., Goulette, F., Gall, Y.L.: Semantic classification of 3d point clouds with multiscale spherical neighborhoods. 3DV pp. 390–398 (2018)

[61] Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and deformable convolution for point clouds. In: ICCV. pp. 6411–6420 (2019)

[62] Toldo, M., Michieli, U., Zanuttigh, P.: Unsupervised domain adaptation in semantic segmentation via orthogonal and clustered embeddings. In: WACV. pp. 1358–1368 (2021)

[63] Wang, H., Shen, T., Zhang, W., Duan, L.Y., Mei, T.: Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In: ECCV (2020)

[64] Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J.: Graph attention convolution for point cloud semantic segmentation. In: CVPR (June 2019)

[65] Wang, S., Zhu, J., Zhang, R.: Meta-rangeseg: Lidar sequence semantic segmentation using multiple feature aggregation. arXiv preprint arXiv:2202.13377 (2022)

[66] Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. TOG **38**(5), 1–12 (2019)

[67] Xiao, A., Yang, X., Lu, S., Guan, D., Huang, J.: Fps-net: A convolutional fusion network for large-scale lidar point cloud segmentation. ISPRS Journal of Photogrammetry and Remote Sensing **176**, 237–249 (2021)

[68] Xie, S., Zheng, Z., Chen, L., Chen, C.: Learning semantic representations for unsupervised domain adaptation. In: ICML (2018)

[69] Yang, B., Wan, F., Liu, C., Li, B., Ji, X., Ye, Q.: Part-based semantic transform for few-shot semantic segmentation. IEEE Transactions on Neural Networks and Learning Systems (2021)

[70] Ye, X., Li, J., Huang, H., Du, L., Zhang, X.: 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In: ECCV. pp. 403–417 (2018)

[71] Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., Tagliasacchi, M.: Soundstream: An end-to-end neural audio codec. IEEE/ACM Transactions on Audio, Speech, and Language Processing **30**, 495–507 (2022). https://doi.org/10.1109/TASLP.2021.3129994

[72] Zhang, R., Zeng, Z., Guo, Z., Gao, X., Fu, K., Shi, J.: Dspoint: Dual-scale point cloud recognition with high-frequency fusion. arXiv preprint arXiv:2111.10332 (2021)

[73] Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In: CVPR. pp. 9601–9610 (2020)

[74] Zhang, Z., Hua, B.S., Yeung, S.K.: Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: ICCV. pp. 1607–1616 (2019)

[75] Zhao, H., Jiang, L., Fu, C.W., Jia, J.: Pointweb: Enhancing local neighborhood features for point cloud processing. In: CVPR. pp. 5565–5573 (2019)

[76] Zhao, H., Jiang, L., Jia, J., Torr, P., Koltun, V.: Point transformer (2021)

[77] Zhao, L., Tao, W.: Jsnet: Joint instance and semantic segmentation of 3d point clouds. In: AAAI (2020)

[78] Zhao, N., Chua, T.S., Lee, G.H.: Few-shot 3D point cloud semantic segmentation. In: CVPR. pp. 8873–8882 (2021)

[79] Zhou, T., Wang, W., Konukoglu, E., Van Gool, L.: Rethinking semantic segmentation: A prototype view. In: CVPR (2022)

[80] Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. CVPR (2021)

[81] Zhu, Y., Zhang, Z., Wu, C., Zhang, Z., He, T., Zhang, H., Manmatha, R., Li, M., Smola, A.: Improving semantic segmentation via self-training. arXiv preprint arXiv:2004.14960 (2020)