UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING
Ph.D. Course in Information Engineering
Information and Communication Science and Technologies Curriculum
XXXVI series

On the Role of Information in Distributed Learning

Ph.D. Candidate
Francesco Pase

Ph.D. Supervisor
Professor Michele Zorzi

Ph.D. Coordinator
Professor Fabio Vandin

Academic Year
2022–2023

To my Mum and Dad, ever-present support and solid home.

"No matter if you're an engineer, a scientist or an artist,
you'll always have to couple the creative world of your mind
with the sharpness of your tools."

Abstract

Today, most data consumed by machine learning algorithms is generated by the enormous amount of sensors and embedded devices like smartphones, cars, drones, which are geographically distributed and potentially concerned about privacy guarantees. In response to this, machine learning systems are expected to transition from centralized to more distributed solutions, in which the training and/or the inference processes are brought closer to the source of the data. In such a setup, the natural learning and inference loops must consider the communication aspect between the involved entities which collaboratively train and run the learning models. In this regard, this work explores the way the exchanged information can be compressed, represented and conveyed through the communication networks; the impacts of unreliable and constrained communication channels on the system outputs; and the fundamental trade-off between the amount of exchanged information and the final performance. Specifically, the analyses first focus on standard federated learning, which is a very popular distributed training technique, and then switch to multi-agent reinforcement learning, in which the underlying learning problem is a decision process. In the end, examples of applications to the optimization of wireless communications networks are also provided.

Contents

Listing of figures

Listing of tables

1 Introduction

The new millennium has witnessed a pervasive adoption of machine learning (ML) in many products and services. Specifically, Deep Learning (DL) is playing a pivotal role in the implementation of many state-of-the-art algorithms: from producing and summarizing text with Large Language Models (LLMs) [183, 239], predicting protein unfolding structures [129], detecting objects in images [63] and playing video games [251], to generating paintings [208], music [68], and many more. Moreover, the prevision on future investments in artificial intelligence (AI) research and startups suggests that the trend is still increasing [1].

Although DL represents a very powerful tool to build the software of the future, to successfully implement such solutions there is the need to collect vast amounts of data, and to have a lot of computational resources at disposal. Until now real-world AI systems have been designed following a centralized approach: data are stored in the cloud where powerful computational infrastructures have the capacity to train huge neural networks (NNs) yielding fully trained models, which are generally run in the cloud also during the inference phase. However, today this type of approach is becoming infeasible for several reasons. First, nowadays the data typically used to train these models are naturally collected by embedded/mobile devices at the edge of the communications network, e.g., sensors, cars, smartphones, drones, and continuously moving the acquired data to the cloud would have a significant impact on the network's resources, as well as on the device's battery. Second, centralized approaches cannot respect, by design, the users' privacy. Third, if inference is performed in the cloud, there will always be a constant and non-negligible delay from the moment the input is acquired by the device to the moment the model produces its output. Indeed, when the inference phase is done in the cloud, the time needed to send the input to the cloud, process it, and send it back to the device cannot be eliminated. However, applications like autonomous driving, virtual reality and remote control can not admit such a delay to safely

[1] https://www.goldmansachs.com/intelligence/pages/ai-investment-forecast-to-approach-200-billion-globally-by-2025.html

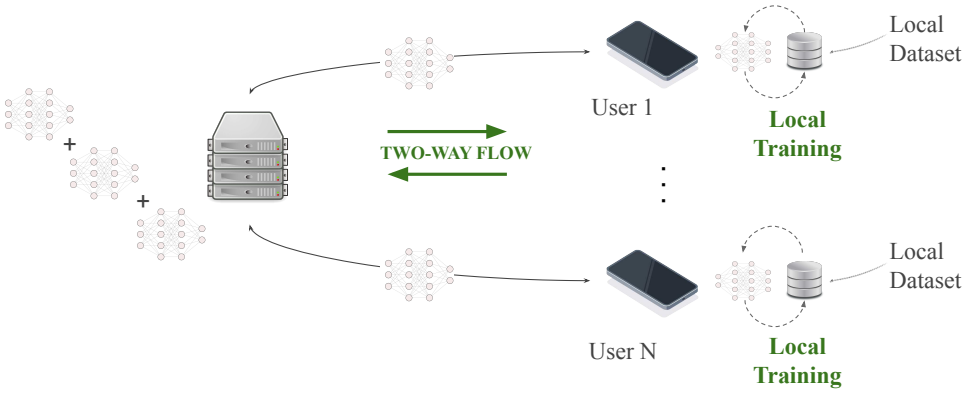


Figure 1.1: General overview of the federated learning framework.

run their operations. Fourth, multi-agent distributed systems, in which a multitude of entities cooperate to share knowledge and to speed up data collection and model training by parallel executions achieving overall better performance, are becoming not only attractive, but sometimes necessary, as suggested by Prof. Geoffrey Hinton in a recent interview [2].

For these reasons, the past few years have experienced a transition from centralized to distributed ML systems, moving not only the inference, but also the training operations, from the cloud to the end devices. Many approaches have been developed in the literature to sustain this paradigm shift, among which federated learning (FL) is now playing a major role [171], and is forseen to become a standard framework for distributed and privacy-preserving training. In short, FL (which will be better defined later) consists in a distributed learning approach where a central authority, i.e., a server or a base station (BS), coordinates the process of training a unique global model among N users (defined also as clients, or in general devices), which are the ones collecting the data (see Figure 1.1). In practice, it works in rounds, and each round consists of the following operations:

1. The server samples a subset of K users (among the N users in the system), and shares with them the global model;

2. Each user trains the received model using their local and potentially private datasets for a number of epoch, i.e., optimization steps, by using some variant of the stochastic gradient descent (SGD) algorithm [25];

3. Each user sends back to the server an updated version of the model;

[2] https://www.youtube.com/watch?v=qpoRO378qRY& ab_channel=CBSMornings

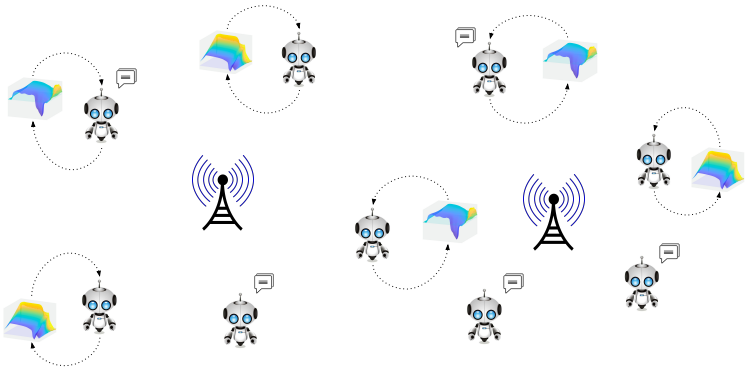


Figure 1.2: A multi-agent learning system.

4. The server receives the updated models, aggregates them (e.g., by averaging) and outputs a new model containing the knowledge of all the users;

5. The process goes back to step 1. and continues until training convergence.

This way, at the end of the process, it is possible to obtain a global model which contains the knowledge of all the users, without having the data to physically leave the users' devices.

Moving one step further, multi-agent distributed systems appear not only in the context of unsupervised and supervised learning, for which FL was initially designed, but also in *interactive learning*, in which agents need to learn from data how to optimize an underlying decision process by interacting (i.e., taking actions) with the process itself. In this case, multi-armed bandit (MAB) [226] and reinforcement learning (RL) [233] are the standard ways of formulating and solving the problem. Even though there already are tentative approaches combining FL with MAB [225, 119] and RL [128], the problem of cooperative learning in multi-agent systems is complex and not so well investigated yet. Examples of applications can be found in the control of wind farm turbines [250], Internet of Things (IoT) [206, 277], wireless networks optimization [186], and many more.

In both types of multi-agent distributed learning systems, the recent literature has usually placed more emphasis on analyzing the optimization, learning convergence, and cooperative challanges of the frameworks [54, 132], sometimes overlooking the communication aspects, which are instead an important part of the problem. The main contribution of this work is to investigate the role of the data exchanged by the entities in distributed learning systems, with the goal of minimizing the amount of information that needs to be communicated through the network to perform distributed learning and inference, sometimes trading off information with performance.

1.1 Communication-Learning Co-Design

As briefly motivated in the previous paragraphs, when designing distributed learning systems it is fundamental to better consider which type of information should be shared, when, and how, in order not to waste communication and storage resources, which sometimes may be scarce. As highlighted later in the manuscript, similar studies naturally lead to questioning the fundamental aspect of information even in centralized settings: is there redundancy in the modern deep neural networks (DNNs)? What is the role of information consumed by the model during training? Is it possible to reduce the size of the models without dramatically decreasing the performance?

In general, when designing the distributed learning systems of the future, it is clear that the network becomes an indivisible part of the system. Indeed, on the one hand the underlying learning process has to consider possibly noisy and/or limited communications among the system's entities, and on the other hand the communication protocols and algorithms should be optimized to consider the specific type of information running through the network, and designed with the final learning goal in mind. Throughout this work we are going to analyze the problem considering different learning tasks, and adopting different perspectives: from model compression and coding [200, 197] to semantic communications [198, 234] and policy compression [187, 188], with applications [194, 249].

1.2 Thesis Organization

The thesis is structured as follows: After this introduction, Chapter 2 focuses on federated learning, with the aim of compressing and representing neural networks to make the framework communication-efficient; Chapter 3 on the other hand considers the communication aspects in the context of distributed decision processes, where the standard frameworks of multi-armed bandit and reinforcement learning are extended by introducing multiple agents and communication channels; Chapter 4 explores applications of distributed learning to wireless network optimization, and to the problem of controlling a swarm of drones; In the end, Chapter 5 summarizes the outcomes of the thesis and draws some final considerations on the future of the field.

2

Information in Federated Learning

In this part of the thesis we are going to analyze the role of information in the specific context of federated learning (FL). Specifically, the focus is on the uplink communication from the clients to the server, in which model or gradient updates are communicated for central aggregation. Consequently, in this context the messages exchanged through the network contain information on the model updates (or even the new model itself). In order to reduce the communication burden, to trade off performance for network resources, several solutions are possible.

First, the server can dynamically adjust the amount of information collected at each round by adapting the number of sampled clients depending on the wireless channel conditions. This is particularly useful when the participating clients are connected to the central authority, e.g., a base station (BS), using the same wireless medium, that is to be shared. In Section 2.2 the impact of the wireless channel on the training convergence is empirically evaluated by running experiments with different channel statistics and conditions, and a way to minimize the negative effects is proposed, which trades off the amount of collected information at each round with its duration.

Moving on, in Section 2.3 a new efficient FL method, called `FedPM`, is proposed. In this case the wireless characteristics are not modeled, but the communication links are abstracted as simple pipes with very limited bandwidth, and we study the performance imposing a communication rate of ~ 1 bit per parameter (bpp). The focus is on reducing the model size, and so on compressing the deep neural networks (DNNs), in order to decrease the amount of necessary physical resources to convey, store, and run the models. To tackle this problem, `FedPM` takes a radically different approach to make the federation process very efficient. Instead of training the actual real-valued coefficients of a neural network, `FedPM` initializes a network randomly (as usual), and then freezes its weights. Then, instead of training the weights, `FedPM` trains a stochastic binary mask whose components indicates which weights should be kept or removed from the original dense model. This way, at the end of the training, an optimized binary mask is obtained, which suggests the

optimal topology of the original model which can better perform on a given problem. With this change in the training perspective, FedPM is able to outperform the state-of-the-art baselines operating at the same communication rate of ~ 1 bpp.

The results obtained in Section 2.3 raise important questions: why are FedPM (and other baselines) still consuming resources when the model has already converged? Why is the rate close to one in the beginning? (see Figure 2.11). Theoretically speaking these observations make sense but in practice it seems we are missing something. Being a stochastic framework, to code the model updates in FedPM clients adopt a standard strategy: they first take a sample of the model, and they then code it by adopting arithmetic coding [69]. In the beginning, no knowledge implies high uncertainty on the model distribution, and so high entropy which induces higher bpp. What is missing in FedPM and in almost all the standard baselines are two main important facts: (i) many training strategies and almost all compression methods are stochastic frameworks. The fact that the model can be expressed as an optimized distribution, in place of a deterministic value, opens the door to a radically new way of representing and communicating model updates in FL; (ii) the side (or prior) information the server has on the model in the form of the global shared model used to initialize each FL round is not typically used in the literature to reduce the rate needed to code the local updates. In Section 2.4 we present KL Minimization with Side Information (KLMS), which optimally exploits these two observations and dramatically reduces the bpp needed to obtain a certain performance guarantee. Specifically, KLMS further optimizes the accuracy-rate trade-off previously obtained by FedPM, pushing once again further the limits of the state of the art.

This Chapter concludes with an attempt to generalize the ideas behind KLMS, proposing a new way of thinking for the general problem of communicating learning models. Even though the system model does not consider the typical FL setting, Section 2.5 provides an information-theoretic perspective on the problem of communicating models, which is clearly of paramount importance for FL, through the lens of semantic communications. The outcomes provide some theoretical guidelines to build the fundamental blocks to design communication schemes which are particularly tailored for running distributed learning algorithms.

2.1 Related Work

In this section, the literature on communication-efficient learning is discussed, covering aspects of FL in wireless networks, model compression, subnetworks (or masks) training, importance sampling as an efficient way to code random samples, and information-theoretic tools for distributed learning.

FL over Wireless Networks. Implementing FL over wireless networks raises several concerns, mainly due to the noisy nature of the wireless links connecting the end devices, as well as the limited computation and communication resources available at each client. Along these lines, Yang *et al.*, in [270], tried to optimize both wireless and computational resources to minimize the learning training delay, even though considering the whole pool of clients at each round. In turn, FL

methods typically select only a subset of devices at each iteration, in order to alleviate the burden of data transmission for distributing model updates. For example, the authors in [26] proposed a method to identify the optimal resource allocation policy as a function of the number of clients participating in the training process, while considering both channel conditions and the significance of their local model updates. The results suggest that the number of clients to be considered at each round should depend on how data are distributed on the local datasets: for independent and identically distributed (i.i.d.) data, the best strategy is to sample just one client per round whereas, for a non-i.i.d. scenario, the number of clients per iteration should be proportional to how heterogeneously the data are distributed, to avoid fitting locally skewed datasets. Another approach to reduce wireless resource occupancy during training is to compress and send sparse local model updates to the server, rather than quantizing the global model itself [27]. Despite these early results, however, it is still not clear, for non-i.i.d. data distributions, how to quantify the trade-off between the number of clients involved in the training of the model and the number of training iterations that are needed to achieve a certain level of accuracy [171].

Compression for FL. There has been extensive research in reducing the communication cost of FL (i) by compressing the model updates through sparsification [20, 37, 159, 184, 256], quantization [174, 232, 253, 262], and low-rank factorization [39, 177, 252, 256]; or (ii) by training sparse subnetworks instead of the full model [200, 154, 153, 166, 178, 243]. Among these approaches, those based on stochastic updates have shown success over the deterministic ones in similar settings. `QSGD` [24] is an effective stochastic quantization method which quantizes model updates into a set of pre-defined and discrete quantization levels – outperforming most other schemes such as `SignSGD` [44] and `TernGrad` [262] by large margins. Lastly, in the Bayesian FL setting, `QLSD` [253] proposes a Bayesian counterpart of `QSGD`, and performs better than other baselines [59, 81, 203].

Pruning for FL. Since the introduction of the Lottery Ticket Hypothesis (LTH) [94], there has been growing interest in finding sparse and *trainable* networks at initialization. The main hypothesis in this line of work is that there exist sparse networks (lottery tickets) inside randomly initialized dense networks such that those sparse networks can be *trained* to a surprisingly good performance – sometimes comparable, if not higher, to the performance of the trained dense network. In the original paper, the strategy for finding these lottery tickets is to iteratively train the dense network until convergence, and so finding the lottery tickets is very expensive. However, the FL papers that utilize the LTH [154, 126, 218] and pruning [157, 179, 273, 166, 127, 32, 75, 157, 46] still present major flaws: (i) These methods require training the weight values, and thus cannot provide an efficient representation of the final model. (ii) Some of these works require finding the lottery tickets prior to FL training [154]. While this could improve the communication cost during the FL training since they communicate sparse networks, it increases the computation cost significantly due to the burden of finding lottery tickets before the federation process starts. (iii) Many of those literature proposals rely on pre-defined sparsity level, whereas Federated Probabilistic Mask Training (`FedPM`) (see Section 2.3) learns with what probability a particular weight should stay in the final model, i.e., the final sparsity level is also a learned

parameter optimized for the best performance.

Finding Subnetworks Inside a Random Network. Regarding mask training, works in [279, 207, 201, 21] find subnetworks (or supermasks) inside a dense network with *random* weights that perform surprisingly well without ever training the weights, but in a centralized scenario. In Section 2.3 we take advantage of the existence of such subnetworks and propose a way to reduce the communication budget in FL to less than 1 bpp with faster convergence and higher accuracy than the relevant baselines in the same bitrate regime, while further compressing the final model, all simultaneously. Prior works [153, 243, 178] also consider finding subnetworks inside a dense random network in a FL setting, but they differ from our approach on several levels. For instance, they focus on different challenges in FL, such as personalization and poisoning attacks, which limits their ability to improve over existing compression methods in accuracy-communication bitrate tradeoff. One fundamental reason for this is their deterministic mask training strategy, which involves hard thresholding or sign operations. On the other hand, the stochasticity in our proposal allows us to (i) enjoy a better accuracy-communication cost tradeoff, (ii) have an unbiased estimate of the true aggregate of the local masks with a provable upper bound on the error, (iii) design an improved aggregation strategy with a Bayesian approach so that the previous masks at the server are not hard replaced – a useful strategy specifically in unbalanced non-IID splits, and (iv) gain privacy benefits via amplification in the Bernoulli sampling step. To demonstrate these benefits over deterministic schemes, we compare our method against FedMask [153] by adapting it slightly to mainly focus on communication efficiency, rather than personalization, and to improve its accuracy-communication efficiency performance.

Importance Sampling Coding. More recently the research community has been looking into the concept of communicating samples [236], which is a very efficient way of conveying some stochastic value, in place of deterministic data (see Section 2.4 for the details). Specifically, in Section 2.4 we present `KLMS`, which adopts the importance sampling algorithm studied in [58, 106, 236], and later applied for model compression [108], learned image compression [89, 90], and compressing differentially private mechanisms [219, 240]. One relevant work is [108], which applies the importance sampling strategy in [58] to compress Bayesian neural networks. Since the model size is too large to be compressed at once, they compress fixed-size blocks of the model parameters separately and independently. As we elaborate in Section 2.4.2, this can be done much more efficiently by choosing the block size adaptively based on the information content of each parameter. While this adaptive strategy could bring some extra communication overhead when applied for model compression (to locate the adaptive-size blocks), we explain how to avoid this overhead in the FL setting by exploiting temporal correlations. Another relevant work is `DP-REC` [240], which again applies the importance sampling technique in [58] to compress the model updates in FL, while also showing differential privacy implications. However, since their training strategy is fully deterministic (no probabilistic learning or stochastic compression), the choice of coding distributions is somewhat arbitrary. Instead, in our work, the goal is to exploit the available side information at the server by choosing natural coding distributions – which improves

the communication efficiency over DP-REC significantly. Another factor in this improvement is the adaptive bit allocation strategy mentioned above – which could actually be integrated into DP-REC as well by avoiding the extra communication overhead as we do in our work (since DP-REC works in an FL setting too). Section 2.4.2 extends also the theoretical guarantees of importance sampling, which quantifies the required bitrate for a target discrepancy (due to compression), to the distributed setting, where we can recover the existing results in [58] as a special case by setting $N = 1$.

Rate-Distortion for Model Communication. In [108] the authors studied the single-shot version of neural networks (NNs) communications, focusing on the design of a practical coding scheme, which is called MIRACLE, to efficiently compress NNs. In [76], the authors study the connections between compressibility and learnability in the context of probably approximately correct (PAC) learning, and show that the two concepts are equivalent when zero/one loss is considered, but not in the case of general loss functions. Another line of research investigates the connections between the generalization capabilities of learning algorithms, and the mutual information between the data and the model [266, 40, 230]. The logic behind these results is to provide a bound on the generalization gap, i.e., the difference between the expected error and the training one, given some information-theoretic properties of the learning algorithm. However, if the environment imposes a constraint on such quantities, e.g., mutual information between the input and output of the learning rule, for example by introducing a rate-limited communication channel between the data and the final model, this influences not only the generalization gap, but also the training error itself (the output of the learning rule is constrained by the environment now), and so it is not clear how the gap between the best achievable test error changes as a function of the mutual information. This is the scenario studied in Section 2.5, where the constraint on the mutual information is not a property of the learning rule, but rather a physical limit imposed by the system. In [187], a similar study is performed on the specific case of contextual multi-armed bandits, where the fundamental quantity is the mutual information $I(S; A)$ between the system states and the action taken by the agents, which is a property of the specific policy adopted. Section 2.5 generalizes that idea to the supervised learning framework, and considers the effect of the communication rate R on the final performance. It is also interesting to highlight the connections between this work and the study in [16], where the authors quantify the complexity of a learning algorithm output Q with its Kullback–Leibler divergence from a prior model distribution P, which, in our system model, represents the minimum achievable rate to convey Q, when P is set as the coding distribution. Section 2.5 is partially built on top of the results in [71, 140, 105, 155, 235], which generalize the concept of rate-distortion theory [69] for standard data communication to probability distributions, where the fidelity requirement at the receiver is not to exactly reconstruct the input data, but rather to generate samples according to some input distribution. Indeed, in the problem investigated in Section 2.5 there is a semantic aspect of communication which is captured by the fact that there is no need to convey the exact data sampled by the transmitter, but rather to represent with high fidelity the belief on the underlying learning model, which is the post-data probability distribution over the class of feasible models.

2.2 Federated Learning over Wireless Networks

2.2.1 Introduction

Artificial intelligence (AI) will play a prominent role in the design and optimization of sixth generation (6G) wireless networks [96]. As previously described, it is envisioned that the co-design of communications systems and applications running on top of them will facilitate an efficient use of wireless physical resources, thereby enabling future vertical services to fulfill very demanding sets of requirements, and FL has gained a lot of interest as a promising and efficient tool to bring intelligence to the edge, where devices collaborate to maintain fresh learning models rather than uploading raw data to centralized servers [171]. However, the underlying unreliable nature of the wireless channel, together with the possibly limited physical network resources available for the federation process, make it sometimes very hard to successfully run FL over real wireless networks. As briefly explained in Section 2.1, it is still not clear, for non-i.i.d. data distributions, how to quantify the trade-off between the number of clients involved in the training of the model and the number of training iterations that are needed to achieve a certain level of accuracy [171]. Moreover, most methods assume perfect knowledge of the Channel State Information (CSI) at each round, which is then leveraged to find the optimal power and resource allocation strategy to minimize the training time. However, perfect CSI may be difficult to obtain in practice, especially in case of fast fading channels. To the best of our knowledge, the only prior work attempting to analyze the FL training process under channel uncertainty is [254], where CSI is inferred using a Gaussian Process (GP) and radio resources are scheduled according to the estimated CSI. However, the analysis does not investigate whether training times are affected by different channel statistics.

Based on this introduction, here we propose a novel FL training method, hereby referred to as Fixed Rate Federated Learning (FRFL), working under imperfect CSI[1], with frequency and time constraints. More specifically, we analyze the convergence time as a function of different channel models (i.e., Rayleigh, Nakagami, and Rician, to characterize different fading regimes), data distributions (i.e., iid and non-iid), and the number of clients participating in the training of the model.

Our contributions can be summarized as follows.

- We evaluate whether exploiting channel statistics, like the cumulative distribution function (CDF) of the fading distribution, when perfect CSI is not available, can still help identify the optimal resource scheduling approach to minimize the convergence time. To do so, we investigate whether preventing clients that cannot sustain a minimum predefined transmission rate from sending model updates results in faster training. Numerical experiments show that, in Rayleigh channels, it is possible to reduce the convergence time by around 80% with 90% accuracy if just half of the clients are able to successfully communicate, compared to a baseline in which all clients adopt the maximum achievable rate to transmit model data.

[1]Unlike in other papers, where the expression *imperfect CSI* denotes the presence of noise or errors in the channel estimation process, here we use it to mean that the only information available about the channel state is its statistical distribution.

- We prove that, as expected, while admitting more clients at each round may not significantly affect the convergence time to achieve a certain accuracy, it can dramatically increase the probability of introducing stragglers into the loop. This effect is particularly remarkable in case of Rayleigh fading, compared to Nakagami and Rician, thus demonstrating how channel statistics should be considered as a bias to optimize scheduling policies for FL.

2.2.2 System Model

In FL, N wireless devices cooperatively build a global model $g(\boldsymbol{\omega})$, stored into a central BS, by sharing learning model updates derived from their local datasets $\mathcal{D}_n$, $n = 1, \ldots, N$, which are a partition of the global dataset $\mathcal{D} = \cup_n \mathcal{D}_n$. The global model parameter vector is randomly initialized to $\boldsymbol{\omega}^0$. The training phase is then organized in rounds, indexed by t. At the beginning of each round, the BS broadcasts the global parameters $\boldsymbol{\omega}^t$ to the clients. Once received, each client n can update its local model $g(\boldsymbol{\omega}_n)$, using a version of the stochastic gradient descent (SGD) algorithm [171], by optimizing its local loss $F_n(g(\boldsymbol{\omega}_n^t), \mathcal{D}_n)$, which is a function of the local model $g(\boldsymbol{\omega}_n^t)$ and its dataset $\mathcal{D}_n$ at round t. At the end of the local optimization phase, the BS selects a pool of C^t clients, with $C^t \leq N$, to collaboratively upload their local model updates, which are then aggregated to generate a new global model that now exploits the knowledge acquired by the clients. The process continues until convergence.

In this work we consider the situation in which the global model $g(\boldsymbol{\omega})$ must be trained within a limited amount of time T, as described in [55]. For example, when a model is used to monitor/control a safety-critical process, e.g., in an Industrial Internet of Things (IIoT) scenario [217] or for teleoperated driving [280], training data must be shared with low latency to guarantee that collaborative machines are synchronized. The problem can then be reformulated as follows. Assume that N wireless devices are connected to the BS using wireless links in an Orthogonal Frequency Division Multiplexing (OFDM) system. At each round, C^t clients are selected and exclusively assigned an orthogonal channel of bandwidth B_k, $k = 1, \ldots, K = C^t \leq N$ [254, 62]. From now on, we will refer to client k as the one associated to the k-th channel. Communication links are modeled as slow fading channels. Unlike previous works, we consider the case in which the BS does not have perfect CSI, but can estimate the CDF $F(h)$ of the channel gain h. In principle, the maximum rate at which client k can communicate its model parameters with arbitrarily low error probability at round t is given by Shannon's formula

$$R_k^t = B_k \log_2 \left(1 + h_k^t \frac{p_k^t \phi_k^t}{N_0 B_k}\right), \quad \forall k \in \{1, \ldots, K\}, \tag{2.1}$$

where h_k^t is the channel gain, p_k^t is the power allocated for transmission, and ϕ_k^t is the path loss experienced by client k during iteration t, whereas N_0 is the noise power spectral density. In our analysis, we consider the case in which clients adapt their power p_k^t in such a way that the path loss and the noise are scaled to reach a constant and target quality factor A, which defines different SNR regimes, i.e.,

$$A = \frac{p_k^t \phi_k^t}{N_0 B_k}, \quad \forall k \in \{1, \ldots, K\}. \tag{2.2}$$

In standard *synchronous* FL, the BS has to wait until all C^t clients involved in the training process at round t upload their local updates before proceeding to the next round, thus the round duration depends on the time required by the slowest client to complete its local computations and update the model. In this work, in turn, we will only consider *communication heterogeneity* in FL [116], and impose that each client performs its local computations within a constant time.

2.2.3 Federated Learning Under Imperfect CSI: The Proposed Solution

As discussed in Sec. 2.2.2, FL methods typically consider a fixed number of clients C^t to be involved in the training phase at round t, and then allocate radio resources in such a way that the time each client takes to upload its model updates within the round is minimized. Different scheduling policies can be adopted depending on whether or not CSI is known a priori. On one side, it is possible to reduce the number of rounds required for convergence by simply increasing C^t. For example, the analysis in [171] shows that, even under non-iid data distribution, increasing at each round the fraction of clients involved in the training process from 10% to 100% could halve the number of training rounds. On the other side, given synchronous FL, the more clients participating at each round, the longer the time required to complete it. Indeed, we can trade the amount of information exchanged at each round, i.e., the client updates, with the total number of rounds that can be completed within a given time T. Notably, if the BS knows the CDF of the channel fading distribution, it is possible to quantify how many model updates from the participating clients can be gathered in T seconds.

Synchronous Federated Learning (SFL) with Perfect CSI

In situations where fresh model updates must be distributed to the edge network with strict time constraints, the BS should accept to complete one round even if some of the clients have yet not shared their federated data, thus increasing the overall number of rounds. In a baseline Synchronous Federated Learning (SFL) approach with perfect channel knowledge, each client k during round t would select the optimal rate to communicate over the channel with arbitrarily low error probability as

$$R_k^t = B_k \log_2(1 + h_k^t A). \tag{2.3}$$

Let Z be the size of the client's local vector parameter $\boldsymbol{\omega}_k$ (which is equal to that of the global model $\boldsymbol{\omega}$), expressed in bits. The time required by client k to reliably transmit the model updates in one round is then given by $T_k^t = Z/R_k^t$, which depends on the specific realization of the channel gain h_k^t, known a priori. With this consideration, we can see that, given the number of clients C^t participating at round t, the round duration T_{round}^t is equal to

$$T_{\text{round}}^t = \max_{k=1,\dots,C^t} \left\{ \frac{Z}{R_k^t} \right\}. \tag{2.4}$$

When C^t is large, T^t_{round} can rapidly grow out of control. Therefore, in practical SFL applications, we shall set $T^t_{\text{round}} \leq T_{\text{ths}}$, so that T^t_{round} never exceeds a predefined threshold T_{ths}.

In this perspective, the rate that dominates the communication delay at round t is determined by $h^t_{\text{m}} = \min_k\{h^t_k\}_{k=1}^{C^t}$, whose CDF and Probability Density Function (PDF) can be found, respectively, as

$$F_{\min}(h^t_{\text{m}}) = 1 - \Big[1 - F(h^t_{\text{m}})\Big]^{C^t}, \tag{2.5}$$

$$f_{\min}(h^t_{\text{m}}) = \frac{\partial F_{\min}(h^t_{\text{m}})}{\partial h^t_{\text{m}}} = C^t\Big[1 - F(h^t_{\text{m}})\Big]^{(C^t-1)} f(h^t_{\text{m}}), \tag{2.6}$$

where $F(h^t_{\text{m}})$ and $f(h^t_{\text{m}})$ are, respectively, the CDF and the PDF of the channel gain h computed in h^t_{m}. The round duration is therefore constrained by the minimum rate $R^t_{\min} = \min_k\{R^t_k\}_{k=1}^{C^t}$, i.e.,

$$T^t_{\text{round}} = \frac{Z}{R^t_{\min}} = \frac{Z}{B_k \log_2(1 + h^t_{\text{m}} A)}. \tag{2.7}$$

Fixed Rate Federated Learning (FRFL) with Imperfect CSI

In this section, we generalize the problem and assume that instantaneous channel information is not available at the server. If CSI is unknown, it is not possible to find the absolute optimal rate to minimize communication errors as in Eq. (2.3). We then propose a Fixed Rate Federated Learning (FRFL) approach in which each client k involved in the training process adopts a constant global rate $R^t_k = R^*,\ \forall k \in \{1, \ldots, K\},\ \forall t$, in such a way that it can complete each training round within $T^t_{\text{round}} = T_{\text{round}} = Z/R^* \leq T_{\text{ths}}$. From communication theory, it is well known that clients can communicate with rate $R^* \leq B_k \log_2(1 + h^t_k A)$ with arbitrarily low error probability. On the contrary, if the rate is such that $R^* > B_k \log_2(1 + h^t_k A)$, e.g., due to near-far effects or in a moving network, the packet error probability may rapidly grow to one, and the client participating in the training may not be able to communicate its model updates successfully. This situation is also known as *deep fading* condition [29]. In this case, the probability that the server loses the model updates sent from client k at round t is given by

$$\begin{aligned} \epsilon(R^*) &= \mathbb{P}[R^* > B_k \log_2(1 + h^t_k A)] \\ &= \mathbb{P}\left[h^t_k < \left(\frac{2^{(R^*/B_k)} - 1}{A}\right)\right] \\ &= F\left(\frac{2^{(R^*/B_k)} - 1}{A}\right). \end{aligned} \tag{2.8}$$

By exploiting the channel statistics, i.e., the CDF $F(h)$ of the fading distribution h, the average number of clients $\hat{C}(R^*)$ successfully participating in each round t when global rate R^* is adopted can be quantified as

$$\hat{C}(R^*) = C^t\Big[1 - \epsilon(R^*)\Big] = C^t\left[1 - F\left(\frac{2^{(R^*/B_k)} - 1}{A}\right)\right], \tag{2.9}$$

where C^t is the original pool of clients selected by the BS to communicate at round t. It appears clear that the choice of the optimal rate R^* dominates the overall training performance. Indeed, R^* can be adapted to include fewer or more clients in the training process, depending on the target number of iterations that must be completed within time T, and the average duration of each round. In FRFL, we adopt a heuristic approach. The BS first computes the expected minimum rate $\mathbb{E}[R^t_{\text{min}}]$ experienced by the C^t participating clients, and then selects R^* such that $R^* > \mathbb{E}[R^t_{\text{min}}]$ if the corresponding error $\epsilon(R^*)$ is below an arbitrary threshold that is deemed acceptably low to allow proper accuracy in the training. By the convexity of the function $\psi(R^t_{min}) = 1/R^t_{\text{min}}$ and Jensen's inequality, it results that $\mathbb{E}[T^t_{\text{round}}] \geq Z/\mathbb{E}[R^t_{\text{min}}]$: using a fixed rate $R^* = \alpha\mathbb{E}[R^t_{\text{min}}]$, with $\alpha > 1$, results in a reduction of the lower bound for the average round duration compared to the baseline SFL method, as expressed in Eq. (2.7), by a factor α, as we will demonstrate in Sec. 2.2.4. We do not preclude more sophisticated methods, e.g., based on mathematical analyses or reinforcement learning, to be adopted for selecting R^*, even though this is out of the scope of this work and will be part of our future work.

2.2.4 Performance Results

In this section, we describe our simulation settings, i.e., the channel models and parameters we adopt, and present our numerical results.

Channel Models

Unlike most literature analyses, in this work we characterize the FL training performance as a function of different channel models, so as to incorporate the effect of different fading regimes.[2] Let $F(h)$ be the CDF of the channel gain h, where in the rest of the analysis we omit indices k and t to indicate the client and the round, respectively, under the assumption that channel realizations are i.i.d. in frequency and time. The following channel models are considered [29].

Rayleigh channel The Rayleigh channel model represents a single diffuse component [79], and is one of the most widely adopted channel models in wireless communications thanks to its simplicity and mathematical tractability. Let σ^2 denote the average squared channel gain, i.e., $\mathbb{E}[h^2] = \sigma^2$; the CDF $F(h)$ of h is then computed as

$$F(h) = 1 - e^{-\frac{h^2}{2\sigma^2}},\ h \geq 0 \tag{2.10}$$

In our experiment, we consider the standard Rayleigh parameterization with $\sigma^2 = 1$, as typically considered in legacy communication systems.

[2]Notice that, while Rayleigh fading is generally assumed for transmissions in the legacy bands, 5G and beyond communication systems may operate in new spectrum bands, e.g., the lower part of the millimeter wave (mmWave) bands [209], where a Rician or Nakagami model would better characterize the effect of multi path components, as expected at those frequencies [150].

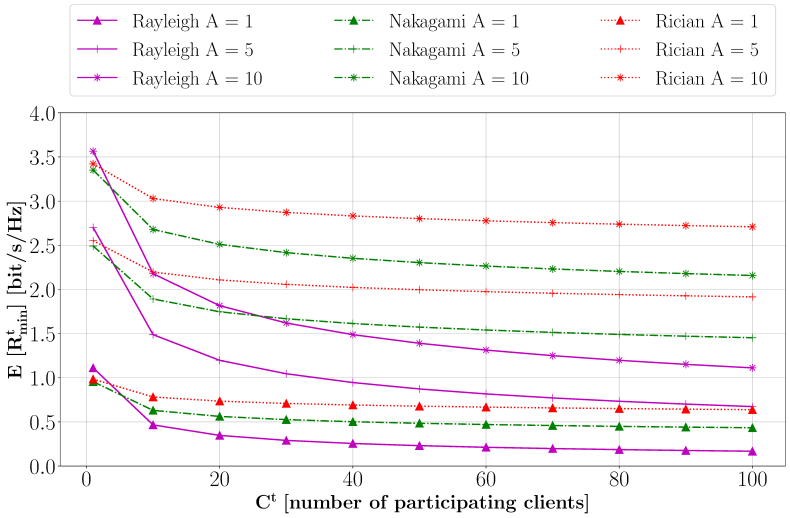


Figure 2.1: Average minimum rate/Hz vs. number of participating clients for the SFL policy in case of Rayleigh, Nakagami and Rician channels and for different values of the quality factor A.

Rician channel The Rician distribution is usually adopted to model an additional dominant, specular, multi path component from the transmitter to the receiver [79]. The channel is parameterized by the factor $K = \nu^2/(2\sigma^2)$, where ν^2 is the contribution of the multi path component power, and σ^2 is related to the diffuse component, as in the Rayleigh case. The CDF $F(h)$ of h is given by

$$F(h) = 1 - Q_1\left(\frac{\nu}{\sigma}, \frac{h}{\sigma}\right), \; h \geq 0 \tag{2.11}$$

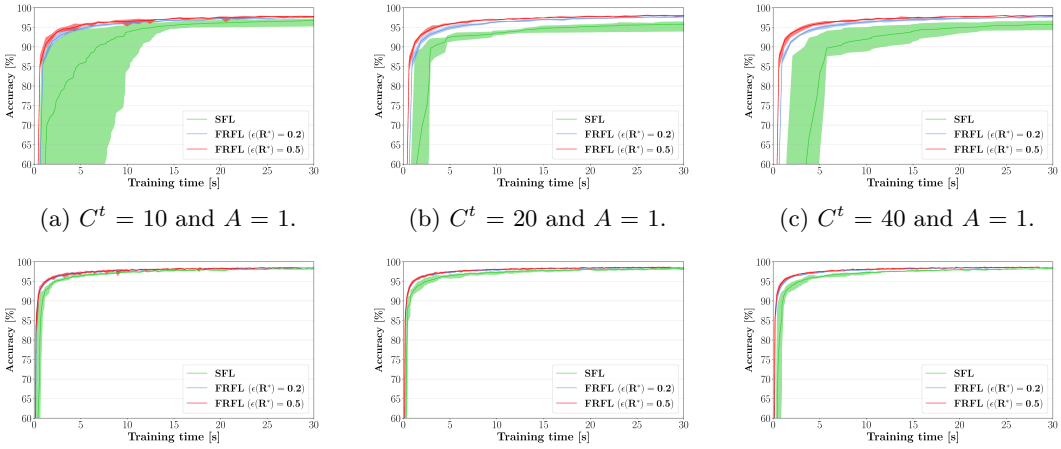
where Q_1 is the Marcum Q-function. We parameterize the Rician model with $K = 12$ dB [215].

Nakagami channel The Nakagami distribution extends the Rayleigh model to incorporate multiple clusters, and is parameterized by the shape parameter m, which represents the number of i.i.d. diffuse components, each modeled as a Rayleigh distribution with mean diffuse power σ^2 [181]. The corresponding CDF $F(h)$ of h is given by

$$F(h) = \frac{\gamma(m, \frac{m}{\sigma^2}h^2)}{\Gamma(m)}, \; h \geq 0 \tag{2.12}$$

where $\gamma(\cdot,\cdot)$ is the lower incomplete Gamma function, and $\Gamma(\cdot)$ is the Gamma function. In this analysis we set $m = 3$ [33].

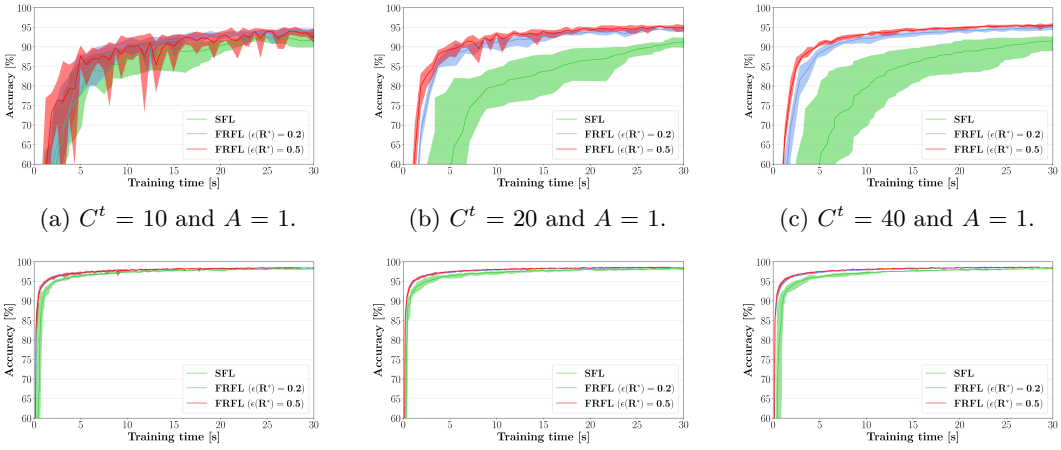
In Fig. 2.1 we plot the average minimum rate $\mathbb{E}[R^t_{\min}]$ for different channel distributions, as a function of the quality factor A and the number of clients C^t that participate in a generic training round t, when perfect CSI is available. We observe that $\mathbb{E}[R^t_{\min}]$ decreases significantly as the number of clients increases, especially when Rayleigh channels are considered. This is expected as the Nakagami and Rician models present a smaller variance. For example, in the presence of



(a) $C^t = 10$ and $A = 1$. (b) $C^t = 20$ and $A = 1$. (c) $C^t = 40$ and $A = 1$.

(d) $C^t = 10$ and $A = 10$. (e) $C^t = 20$ and $A = 10$. (f) $C^t = 40$ and $A = 10$.

Figure 2.2: Min-to-max and average accuracy (over 5 simulations) during the training process as a function of the time and the number of clients C^t involved in the rounds, considering both SFL and FRFL methods. Rayleigh fading with $A = 1$ (first row) and $A = 10$ (second row), and i.i.d. data are considered.



(a) $C^t = 10$ and $A = 1$. (b) $C^t = 20$ and $A = 1$. (c) $C^t = 40$ and $A = 1$.

(d) $C^t = 10$ and $A = 10$. (e) $C^t = 20$ and $A = 10$. (f) $C^t = 40$ and $A = 10$.

Figure 2.3: Min-to-max and average accuracy (over 5 simulations) during the training process as a function of the time and the number of clients C^t involved in the rounds, considering both SFL and FRFL methods. Rayleigh fading with $A = 1$ (first row) and $A = 10$ (second row), and non-i.i.d. data are considered.

poor Rayleigh channel conditions, e.g., $A = 1$, the average minimum rate drops by more than 50%, resulting in more than twice the training delay, when only 10 clients are involved in each round. The same effect is observed even in case of strong channels, i.e., $A = 10$, and if 40 clients selected to participate.

Simulation Parameters and Setting

Based on the results in Fig. 2.1, in our simulations we consider $N = 100$ overall wireless clients, while only $C^t \in \{10, 20, 40\}$ of them are selected to participate in the model updates at generic round t. Each participating client uses an orthogonal channel of 1 MHz of bandwidth in all the investigated configurations. Two different values of A, i.e., 1 and 10, are considered in the Rayleigh case, with iid and non-iid data distributions, whereas $A = 1$ is selected for Rician and Nakagami channels. In our experiments we evaluate the performance of the FL training process, specifically the convergence time, comparing two different scheduling strategies: a baseline SFL approach with full channel information, and two different versions of the FRFL strategy working under imperfect CSI, with $\epsilon(R^*) = 0.2$ and 0.5. The two models assume that on average 20% and 50% of the clients, respectively, are not able to communicate their training updates due to bad channel conditions at the selected global rate R^*. In both cases, $\epsilon(R^*)$ has been selected so that $R^* > \mathbb{E}[R^t_{\min}]$ in all simulation scenarios. The training time is set to $T = 30$ seconds, which is large enough to let the model be trained with an acceptable level of accuracy.

The simulations are conducted on the MNIST dataset [271], which contains 70 000 (60 000 for training and 10 000 for testing) handwritten digits, classified into one of 10 possible classes. While, for i.i.d. data distribution, each client has 600 training samples, and classes are uniformly distributed among the local datasets, in the non-iid setting a random number of training samples and classes are distributed among the clients.

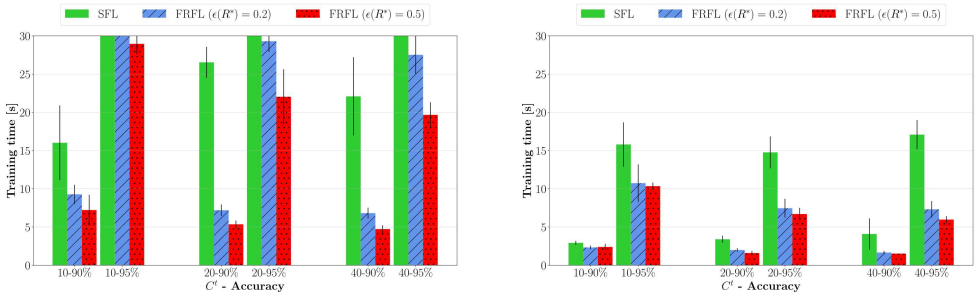
The learning model is a Convolutional Neural Network (CNN) with two 5×5 convolutional layers (with 10 and 20 channels and a 2×2 max pooling operation after the first layer), followed by one dense layer with 320 neurons and one output layer with 10 units. The activation function for the inner layers is the ReLu function, whereas softmax is used for the output layer. The loss is modeled by the cross-entropy function, which is a standard option in classification problems. Training weights are aggregated at the BS according to the FedAvg aggregator function [171]: at the end of round t, the new global vector parameter $\boldsymbol{\omega}^{t+1}$ is computed as

$$\boldsymbol{\omega}^{t+1} = \frac{1}{D^t} \sum_{k=1}^{C^t} D_k^t \boldsymbol{\omega}_k^t, \tag{2.13}$$

where $D^t = \sum_{k=1}^{C^t} D_k^t$, with D_k^t being the size of the local dataset $\mathcal{D}_k^t$, and the local parameter vectors $\{\boldsymbol{\omega}_k^t\}_{k=1}^{C^t}$ are updated using the SGD algorithm with momentum equal to 0.5 and learning rate set to 0.01. Notice that, in FRFL, some clients may not be able to share their local parameter vectors. Therefore, if client k experiences a transmission error during round t, $\boldsymbol{\omega}_k^t$ is set to $\mathbf{0}$ at the BS, and $D_k^t = 0$.

Numerical Results

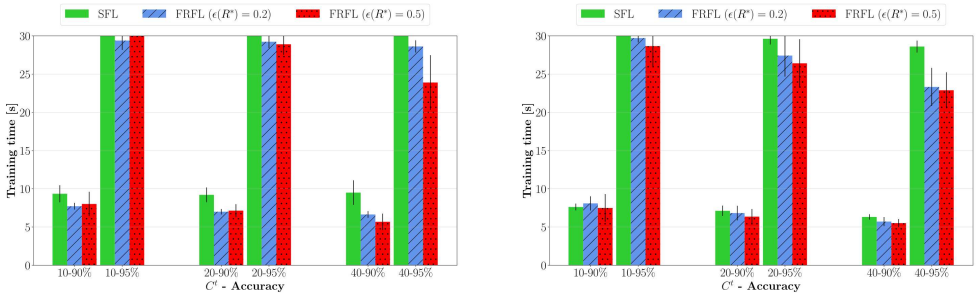
In this section we validate the performance of the proposed FRFL method when imperfect CSI is considered. Fig. 2.2 plots the average accuracy over time achieved on the test dataset during the federated training process in Rayleigh channels, as a function of the number of clients C^t involved



(a) Chanel quality factor $A = 1$.

(b) Chanel quality factor $A = 10$.

Figure 2.4: Average time (and confidence intervals) to achieve 90% and 95% accuracy in Rayleigh fading channels, as a function of the number of clients C^t involved in the rounds, considering both SFL and FRFL methods.



(a) Nakagami channel and $A = 1$.

(b) Rician channel and $A = 1$.

Figure 2.5: Average time (and confidence intervals) to achieve 90% and 95% accuracy in Nakagami and Rician fading channels, as a function of the number of clients C^t involved in the rounds, considering both SFL and FRFL methods.

in the training and the channel condition A, and assuming i.i.d. data.

First, we observe that adding more clients per round does not impact the long-term accuracy even with imperfect CSI, as acknowledged by prior analyses, e.g., in [26]. In fact, FRFL assumes a fixed global rate R^* for all participating clients, which does not affect the transmission delay. On the contrary, in case CSI is available, SFL implies that the more clients involved in the communications rounds, the longer, on average, the time it takes for the server to receive all model updates, which results in slower convergence. For example, at 5 seconds, the accuracy drops from around 95% to 85% when SFL is considered, for $C^t = 10$ and $A = 1$. Fig. 2.2a, Fig. 2.2b, and Fig. 2.2c further demonstrate that considering a weaker channel, i.e., $A = 1$, degrades the long–term accuracy performance of the training, as adding more clients slows down the communications rounds. In case of more robust channels with $A = 10$ (Fig. 2.2d, Fig. 2.2e, and Fig. 2.2f) this effect is mitigated, e.g., at 5 seconds, for $C^t = 10$, the SFL training accuracy increases by around 13% compared to $A = 1$. In any case, FRFL always outperforms SFL, even in the presence of perfect CSI.

In Fig. 2.3, the SFL vs. FRFL performance is evaluated with non-i.i.d. data. In this case,

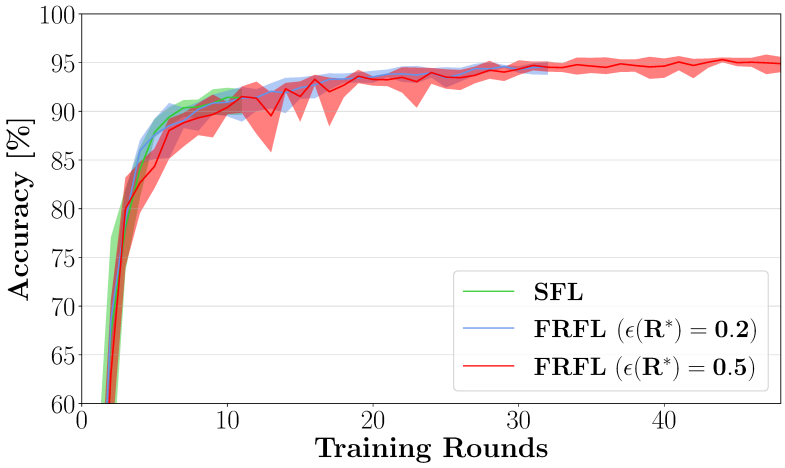


Figure 2.6: Min-to-max and average accuracy (over 5 simulations) during the training process as a function of the number of rounds with Rayleigh fading ($A = 1$), $C^t = 20$, and non-i.i.d. data.

gathering information from a smaller fraction of clients cannot generally sustain sufficiently high levels of accuracy. For example, Fig. 2.3a presents an accuracy always lower than 95% for $C^t = 10$ in all investigated configurations. Increasing the number of clients may improve the accuracy performance during the whole training time, even though this effect is mitigated in the SFL strategy as the more participating clients imply also longer round durations. Moreover, it is interesting to compare the results for the SFL policy with $C^t = 20$ (Fig. 2.3b) and the FRFL policy with $C^t = 40$ and $\epsilon(R^*) = 0.5$ (Fig. 2.3c). In both cases, the training involves 20 participating clients, as FRFL implies that, on average, 50% of the clients do not successfully deliver their model updates on time, i.e., $\hat{C}(R^*) = 20$. Then, even though the FRFL approach achieves better accuracy than SFL despite imperfect CSI (i.e., 95% vs. 90% at the end of the training when $A = 1$), it requires 40 channels to be allocated to the $C^t = 40$ clients, thus consuming twice the frequency resources. However, better performance against SFL can still be guaranteed with $C^t = 20$, that in turn requires 20 orthogonal channels for both policies.

Fig. 2.4 compares the training time required to obtain 90% and 95% accuracy in Rayleigh channels with $A = 1$ and $A = 10$, when either SFL or FRFL is considered, as a function of the number of clients involved in the rounds. First, we observe that it is possible to converge faster by trading the amount of information collected at each round with the round duration, which in turn increases the total number of possible rounds within $T = 30$ s. For example, Fig. 2.4a shows that, when $A = 1$ and $C^t = 20$, the training time to reach 90% accuracy can be reduced by almost 80% if the proposed FRFL training method is adopted. Moreover, when $A = 1$, the baseline SFL configuration, which always tends to assign the largest possible rate to its participating clients, is never able to reach 95% accuracy within the training time despite leveraging full CSI. In turn, the FRFL policy with $\epsilon(R^*) = 0.5$ and $C^t = 40$ succeeds in only 20 seconds, on average, with small deviations. The same conclusions can be derived from Fig. 2.5, which investigates the

impact of different channel models, i.e., Nakagami (Fig. 2.5a) and Rician (Fig. 2.5b), on the convergence time, for $A = 1$. First, we notice that, even though Rayleigh channels guarantee, on average, higher gains in single-link communications, Nakagami and Rician channels can support faster convergence for both SFL and FRFL policies: with Rician fading, for $C^t = 10$, SFL with perfect CSI obtains 95% accuracy in less than 10 seconds, against the 16 seconds when Rayleigh is adopted. This can be explained by the fact that both Nakagami and Rician fading exhibit lower variance, and can admit more clients per round, without increasing the average delay considerably. Nevertheless, the proposed FRFL policy always achieves faster convergence even with imperfect CSI by configuring faster rounds. Finally, Fig. 2.6 depicts the training accuracy as a function of the number of rounds, in case of Rayleigh fading with $A = 1$, $C^t = 20$, and non-iid data. It is possible to see that, within the allocated time $T = 30$ s, the FRFL policy with $\epsilon(R^*) = 0.2$ ($\epsilon(R^*) = 0.5$) is able to operate though 32 (48) rounds, while in turn the SFL policy is limited to 12 rounds, and never achieves 95% accuracy. As a consequence, our analysis demonstrates that it may be convenient to neglect model updates from some participating clients, e.g., the most channel-constrained devices, as per the FRFL strategy, in favor of more round opportunities during training. The same trend is illustrated in Fig. 2.4a with $C^t = 20$.

2.2.5 Conclusions and Future Works

In this work we propose a novel federated learning method that decreases the convergence time by assigning a global constant rate to all the clients participating in the training rounds. Notably, the proposed approach does not require CSI availability, unlike most existing analyses. Our simulation results, validated in different channel regimes, demonstrate that the proposed approach, despite considering imperfect CSI, always achieves better training performance compared to a baseline strategy in which the clients always adopt the maximum achievable rate to transmit model data.

2.3 Sparse Random Networks for Communication-Efficient Federated Learning

2.3.1 Introduction

While being an appealing approach for enabling model training without the need to collect client data at the server, *uplink* communication of local updates is a significant bottleneck in FL [132]. This has motivated research in communication-efficient FL strategies [171] and various gradient compression schemes via sparsification [159, 256, 37, 184, 123], quantization [24, 262, 44, 174], and low-rank approximation [138, 246, 247, 39], as introduced in Section 2.1. In this new proposal, while aiming for communication efficiency in FL, we take a radically different approach from prior work, and propose a strategy that does not require communication of weight updates. To be more precise, instead of training the weights,

(1) the server initializes a dense random network with d weights, denoted by the weight vector $\boldsymbol{w}^{\text{init}} = (w_1^{\text{init}}, w_2^{\text{init}}, \ldots, w_d^{\text{init}})$, using a random seed SEED, and broadcasts SEED to the clients

enabling them to reproduce the same $\boldsymbol{w}^{\text{init}}$ locally,

(2) both the server and the clients keep the weights frozen at their initial values $\boldsymbol{w}^{\text{init}}$ at all times,

(3) clients collaboratively train a *probability mask* of d parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d) \in [0,1]^d$,

(4) the server samples a binary mask from the trained probability mask and generates a sparse network with random weights – or a subnetwork inside the initial dense random network as follows

$$\boldsymbol{w}^{\text{final}} = \text{Bern}(\boldsymbol{\theta}) \odot \boldsymbol{w}^{\text{init}}, \tag{2.14}$$

where $\text{Bern}(\cdot)$ is the Bernoulli sampling operation and $\odot$ the element-wise multiplication.

We call the proposed framework `FedPM` and summarize it in Figure 2.7. At first glance, it may seem surprising that there exist subnetworks inside randomly initialized networks that could perform well without ever modifying the weight values. This phenomenon has been explored to some extent in prior work [279, 207, 201, 78, 21] with different strategies for finding the subnetworks. However, how to find these subnetworks in a FL setting has not attracted much attention so far. Some exceptions to this are works in [153, 243, 178], which provide improvements in other FL challenges, such as personalization and poisoning attacks, while not being competitive with existing (dense) compression methods such as QSGD [24], DRIVE [246], and SignSGD [44] in terms of *accuracy* under the same communication budget. In this work, we propose a *stochastic* way of finding such subnetworks while reaching higher accuracy at a reduced communication cost – less than 1 bit per parameter (bpp).

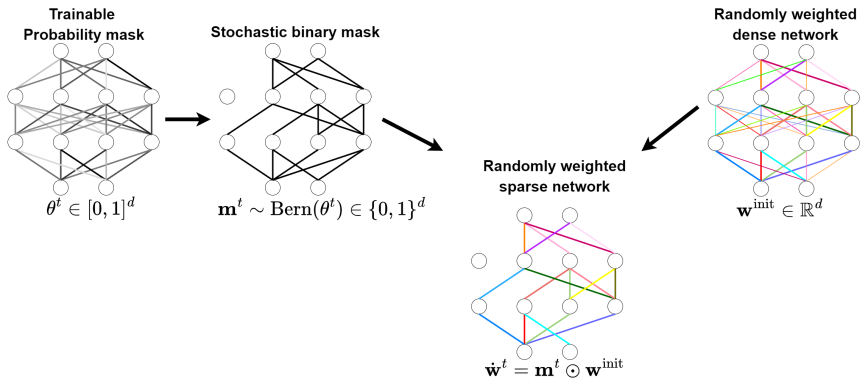


Figure 2.7: Extracting a randomly weighted sparse network using the trainable probability mask $\boldsymbol{\theta}^t$ in the forward-pass of round t (for clients and the server). In practice, clients collaboratively train continuous scores $\boldsymbol{s} \in \mathbb{R}^d$, and then at inference time, the clients (or the server) find $\boldsymbol{\theta}^t = \text{Sigmoid}(\boldsymbol{s}^t) \in [0,1]^d$. We skip this step in the figure for the sake of simplicity.

In addition to the accuracy and communication gains, our framework also provides an efficient representation of the final model post-training by requiring less than 1 bpp to represent (i) the random seed that generates the initial weights $\boldsymbol{w}^{\text{init}}$, and (ii) a sampled binary vector $\text{Bern}(\boldsymbol{\theta})$ (computed with the trained $\boldsymbol{\theta}$). Therefore, the final model enjoys a memory-efficient deployment

– a crucial feature for machine learning at power-constrained edge devices. Another advantage our framework brings is the privacy amplification under some settings, thanks to the stochastic nature of our training strategy.

Our contributions can be summarized as follows:

(1) We propose a FL framework, in which the clients do not train the model weights, but instead a stochastic binary mask to be used in sparsifying the dense network with random weights. This differs from the standard training approaches in the literature.

(2) Our framework provides efficient communication from clients to the server by requiring (less than) 1 bpp per client while yielding faster convergence and higher accuracy than the baselines.

(3) We propose a Bayesian aggregation strategy at the server side to better deal with partial client participation and non-IID data splits.

(4) The final model (a sparse network with random weights) can be efficiently represented with a random seed and a binary mask which requires (less than) 1 bpp – at least 32× more efficient storage and communication of the final model with respect to standard FL strategies.

(5) We demonstrate the efficacy of our strategy on MNIST, EMNSIT, CIFAR-10, and CIFAR-100 datasets under both IID and non-IID data splits; and show improvements in accuracy, bitrate, convergence speed, and final model size over relevant baselines, under various system configurations.

2.3.2 Federated Probabilistic Mask Training (FedPM)

We first describe the simpler version of the FedPM framework, which provides an *unbiased* estimation of the mean of the learned probability masks at the server with *bounded error*. Next, we propose a modification in our aggregation strategy by exploiting the underlying Bernoulli mechanism. This helps boosting the performance of FedPM in the case of partial client participation. We then discuss the details of the distribution of the initial weights, and finally describe the privacy benefits of FedPM. We use capital letters for random variables, small letters for their realization and deterministic quantities, and bold letters for vectors. Moreover, we indicate with $\boldsymbol{x}^{u,t}$ the state of the local vector $\boldsymbol{x}$ (e.g., the local mask) at client u during round t, and with $x_i^{u,t}$ its i-th component. Global values are denoted with $\boldsymbol{x}^{g,t}$ and $x_i^{g,t}$, and sets are indicated with calligraphic fonts. We denote a neural network with weight vector $\boldsymbol{p}$ as $f_{\boldsymbol{p}}$.

FedPM

In this section, we present the general FedPM training pipeline. First, the server randomly initializes a neural network $f_{\boldsymbol{w}^{\text{init}}}$, parameterized by the weight vector $\boldsymbol{w}^{\text{init}} = (w_1^{\text{init}}, w_2^{\text{init}}, \dots, w_d^{\text{init}}) \in \mathbb{R}^d$, whose components are sampled IID according to a distribution $P_{\boldsymbol{w}}$ using a *randomly generated* seed SEED. The random SEED value is then communicated to all the clients, which can locally sample the same pseudo-random vector $\boldsymbol{w}^{\text{init}}$, which is kept fixed and never modified during training. The goal for the clients is to collaboratively train a probability mask $\boldsymbol{\theta} \in [0,1]^d$, which indicates the Bernoulli parameters for the global stochastic binary mask $\boldsymbol{M} \sim \text{Bern}(\boldsymbol{\theta}) \in \{0,1\}^d$, such that the function $f_{\dot{\boldsymbol{W}}}$ maximizes its performance on a given task, where $\dot{\boldsymbol{W}} = \boldsymbol{M} \odot \boldsymbol{w}^{\text{init}}$. Specifically,

FedPM learns the probabilities for the weights of being active, which are given by the probability mask $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d) \in [0,1]^d$. To achieve this, at every round t, the server samples a set $\mathcal{K}_t$ of $|\mathcal{K}_t| = K$ participants (out of the total N clients), which individually train their local probability masks $\boldsymbol{\theta}^{k,t}, k \in \mathcal{K}_t$, by using their local datasets $\mathcal{D}_k$, each composed of $D_k = |\mathcal{D}_k|$ samples. These local masks are then aggregated by the server in a communication-efficient way to estimate the optimal $\boldsymbol{\theta}$. At test time, at the server, the initial random network $f_{\boldsymbol{w}^{\text{init}}}$ is sparsified using the global probability mask $\boldsymbol{\theta}^{g,t}$, following the stochastic approach in Figure 2.7. In the following sections, we provide more details on each step of each round. We give the pseudocode for FedPM in Appendix 2.7.1.

Local Training of Probability Masks

Upon receiving a global probability mask $\boldsymbol{\theta}^{g,t-1}$ from the server at the beginning of round t, the client k performs local training and updates the mask via back-propagation. First, however, we have to guarantee that the updated probability mask satisfies $\boldsymbol{\theta}^{k,t} \in [0,1]^d$. While this can be achieved with a regularization term in the loss, this may require clipping $\boldsymbol{\theta}^{k,t} \in [0,1]^d$ before taking a Bernoulli sample, especially in the early training stages. Clipping would then make the estimate at the server biased and hence lead to a slower convergence and lower accuracy. Therefore, similarly to the work in [279], we introduce another mask, called *score mask* $\boldsymbol{s} = (s_1, s_2, \ldots, s_d) \in \mathbb{R}^d$, that has unbounded support and can be used to generate the probability masks through the one-to-one sigmoid function by setting $\boldsymbol{\theta} = \text{Sigmoid}(\boldsymbol{s})$. Then, the procedure for local training of the probability mask at round t is as follows (here, the steps from Step 2 to 4 describe one local iteration, which is repeated a number τ of times as standard in FL [171]):

(1) The server sends the global probability mask $\boldsymbol{\theta}^{g,t-1}$ to K chosen clients, and the clients set $\boldsymbol{s}^{k,t} = \text{Sigmoid}^{-1}(\boldsymbol{\theta}^{g,t-1})$, where $\text{Sigmoid}^{-1}(\cdot)$ is the inverse of the sigmoid function.

(2) Then, the clients generate a binary mask by first transforming back $\boldsymbol{\theta}^{k,t} = \text{Sigmoid}(\boldsymbol{s}^{k,t})$, and then sampling a binary mask $\boldsymbol{M}^{k,t}$ from $\boldsymbol{\theta}^{k,t}$ as shown in Figure 2.7: $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$.

(3) The sampled binary mask then sparsifies the initial weight vector $\boldsymbol{w}^{\text{init}}$: $\dot{\boldsymbol{w}}^{k,t} = \boldsymbol{m}^{k,t} \odot \boldsymbol{w}^{\text{init}}$.

(4) $\dot{\boldsymbol{w}}^{k,t}$ is then used for forward pass, and the loss $\mathcal{L}(f_{\dot{\boldsymbol{w}}^{k,t}}, \mathcal{D}_k)$ on the local task is backpropagated to update the score mask as $\boldsymbol{s}^{k,t} = \boldsymbol{s}^{k,t} - \eta \nabla \mathcal{L}(f_{\dot{\boldsymbol{w}}^{k,t}}, \mathcal{D}_k)$ (η is the local learning rate).

All the local operations from Step 2 to Step 4 are differentiable, except for the Bernoulli sampling. We backpropagate the gradients through the Bernoulli sampling operation with a straight-through estimator [43], using the first-order gradient of the Bernoulli function, which is simply equal to the probability mask $\boldsymbol{\theta}^{k,t}$.

Communication Strategy

Once the local training at round t is completed, the server needs to distill the global probability mask $\boldsymbol{\theta}^{g,t}$, by taking the empirical average of the local probability masks $\bar{\boldsymbol{\theta}}^{g,t} = \frac{1}{K} \sum_{k \in \mathcal{K}_t} \boldsymbol{\theta}^{k,t}$ collected from the clients. However, since we aim for communication efficiency, the clients do not send their local probability masks directly. Instead, they communicate a stochastic binary sample

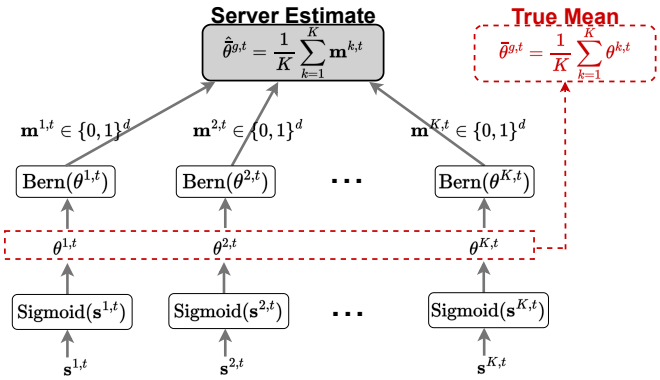


Figure 2.8: Communication-efficient estimation of the mean of the probability masks $\bar{\boldsymbol{\theta}}^{g,t}$. Each client communicates a stochastic binary mask $\boldsymbol{m}^{k,t}$ sampled from the local probabillity mask $\boldsymbol{\theta}^{k,t}$. We reduce the bitrate to less than 1 bit per parameter by using arithmetic coding to encode $\boldsymbol{m}^{k,t}$. When the frequency of 1's is far from 0.5 (which is usually the case with `FedPM`), the number of bits per parameter to communicate $\boldsymbol{m}^{k,t}$ is less than 1. See Figure 2.11 for more details.

$\boldsymbol{M}^{k,t}$ from their probability masks sampled as $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$, and then the server estimates the global aggregate $\bar{\boldsymbol{\theta}}^{g,t}$ as $\hat{\bar{\boldsymbol{\theta}}}^{g,t} = \frac{1}{K}\sum_{k\in\mathcal{K}_t} \boldsymbol{m}^{k,t}$. This distributed mean estimation problem with communication constraints is summarized in Figure 2.8. Our estimator $\hat{\bar{\boldsymbol{\theta}}}^{g,t} = \frac{1}{K}\sum_{k\in\mathcal{K}_t} \boldsymbol{m}^{k,t}$ is an unbiased estimate of the true aggregate, in that

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{M}^{k,t}\sim\text{Bern}(\boldsymbol{\theta}^{k,t})\ \forall k\in\mathcal{K}_t}[\hat{\bar{\boldsymbol{\theta}}}^{g,t}] &= \mathbb{E}_{\boldsymbol{M}^{k,t}\sim\text{Bern}(\boldsymbol{\theta}^{k,t})\ \forall k\in\mathcal{K}_t}\left[\frac{1}{K}\sum_{k\in\mathcal{K}_t}\boldsymbol{M}^{k,t}\right] \\
&= \frac{1}{K}\sum_{k\in\mathcal{K}_t}\mathbb{E}_{\boldsymbol{M}^{k,t}\sim\text{Bern}(\boldsymbol{\theta}^{k,t})}[M^{k,t}] \\
&= \frac{1}{K}\sum_{k\in\mathcal{K}_t}\boldsymbol{\theta}^{k,t} \\
&= \bar{\boldsymbol{\theta}}^{g,t}.
\end{aligned}
$$

Moreover, the estimation error is upper bounded as (the proof is given in Appendix 2.7.2)

$$
\mathbb{E}_{\boldsymbol{M}^{k,t}\sim\text{Bern}(\boldsymbol{\theta}^{k,t})\ \forall k\in\mathcal{K}_t}\left[||\hat{\bar{\boldsymbol{\theta}}}^{g,t} - \bar{\boldsymbol{\theta}}^{g,t}||_2^2\right] \leq \frac{d}{4K}. \tag{2.15}
$$

Since each client communicates a stochastic binary mask $\boldsymbol{M}^{k,t}$, 1 bpp is the worst-case bitrate for `FedPM`. We can further reduce the bitrate to less than 1 by using arithmetic coding [211] or universal coding [141, 38] to encode $\boldsymbol{m}^{k,t}$, and achieve the empirical entropy since d is large. This gives us smaller bitrates whenever the frequency of 1's in $\boldsymbol{m}^{k,t}$ is far from 0.5 – which is usually the case for our method (see Figure 2.11). We note that, with a deterministic mask training approach as in FedMask [153], arithmetic coding of $\boldsymbol{m}^{k,t}$s does not provide any further gain in bitrate, as

we have empirically observed that the frequency of 1's is always around 0.5 (see Figure 2.11 and Table 2.3) – here we apply arithmetic coding for FedMask to improve our baseline although it was not proposed in the original paper. Moreover, FedMask [153] and HideNSeek [243] do not enjoy the guarantees we have as their estimator (i) is not unbiased and (ii) does not have an upper bound on the estimation error due to hard thresholding [153] and sign operations [243]. This is another benefit of our stochastic sampling approach.

`FedPM` with Bayesian Aggregation

Another important aspect that differentiates our work from existing masking methods such as FedMask [153] and HideNSeek [243] is the Bayesian aggregation strategy, which exploits the underlying *stochastic mask* to synthesize a global model, boosting the performance in scenarios where only a fraction of the clients participate in each round. Given the probabilistic interpretation of the `FedPM` mask's values, at the server side we further model the probability mask $\boldsymbol{\theta}^{g,t}$ with a Beta distribution $\text{Beta}(\boldsymbol{\alpha}^{g,t}, \boldsymbol{\beta}^{g,t})$, parameterized by the round-dependent parameters $\boldsymbol{\alpha}^{g,t}$ and $\boldsymbol{\beta}^{g,t}$, which are initialized to $\boldsymbol{\alpha}^{g,0} = \boldsymbol{\beta}^{g,0} = \boldsymbol{\lambda_0}$. At the beginning of the training process, there is no prior knowledge indicating which network weight should be more important than the others, and so each entry in the probability mask is uniformly distributed in $[0, 1]$ – which is the *prior* distribution. Consequently, the clients' local binary masks $\boldsymbol{M}^{k,t}$s are the *data* the server uses to update its belief on each weight score, and so the aggregation strategy corresponds now to a *posterior update*. Specifically, given the conjugate relation between the Beta-Bernoulli distributions, the new posteriors are still Beta distributions with parameters

$$\boldsymbol{\alpha}^{g,t} = \boldsymbol{\alpha}^{g,t-1} + \boldsymbol{M}^{\text{agg},t} \quad \text{and} \quad \boldsymbol{\beta}^{g,t} = \boldsymbol{\beta}^{g,t-1} + K \cdot \mathbf{1} - \boldsymbol{M}^{\text{agg},t} \quad \forall t \geq 1, \tag{2.16}$$

where $\boldsymbol{M}^{\text{agg},t} = \sum_{k \in \mathcal{K}_t} \boldsymbol{M}^{k,t}$, and $\mathbf{1}$ is the d-dimensional all-ones vector. Then, the server broadcasts to the clients the mode of the Bernoulli distributions, as suggested by [88],

$$\boldsymbol{\theta}^{g,t} = \frac{\boldsymbol{\alpha}^{g,t} - 1}{\boldsymbol{\alpha}^{g,t} + \boldsymbol{\beta}^{g,t} - 2}, \tag{2.17}$$

where the division operation is applied element-wise. However, to obtain the best performance out of this method, the Beta parameters should be re-initialized to their original values $\boldsymbol{\lambda}_0$ with some regularity. We present an ablation study to demonstrate the improvements gained by the Bayesian aggregation strategy and the reasonable choices for the resetting frequency in Section 2.3.4. Notice that if $\boldsymbol{\lambda}_0 = \mathbf{1}$, and if $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are re-initialized at the beginning of each round, the method is equivalent to the aggregation strategy detailed in Fig. 2.8.

Weight Distribution

The fixed weight vector $\boldsymbol{w}^{\text{init}}$ is initialized by sampling from the distribution P_w using the randomly generated `SEED`. We note that the choice of this distribution impacts two important aspects of `FedPM`: (i) the values of $\boldsymbol{w}^{\text{init}}$ highly influence the final accuracy achieved by the model, as they

represent the building blocks to extract a subnetwork $f_{\dot{\boldsymbol{w}}}$ (see Figure 2.7), which should be rich enough to solve the learning task, and (ii) the size of the sample space of P_w affects the number of bits needed to store the model during the *inference process* (this is different from the 1 bpp model storage when the model is not in use). Regarding (i), as also proposed in [207], we sample weights from a uniform distribution, whose domain is $\{-\sigma, +\sigma\}$, where σ is the standard deviation of the Kaiming Normal distribution [112]. In this way, we control the variance of the neurons' output to be ~ 1, which avoids the vanishing or the explosion of activation values. Previous experiments in [279, 207] also demonstrate the superior performance achieved by binary weights distributions when compared to standard continuous counterparts, e.g., Gaussian. Regarding (ii), even if knowing the value of SEED is enough to perfectly reconstruct the vector $\boldsymbol{w}^{\text{init}}$, one would have to generate the entire vector at every inference step. Consequently, to achieve fast inference, the actual values of the weights need to be stored in the memory of the devices during the *inference process*. Fortunately, our initialization allows for efficient storage even during inference since (after reconstructing $\boldsymbol{w}^{\text{final}}$ using SEED and $\boldsymbol{m}^{\text{final}} \in \{0,1\}^d$) we only need to indicate whether the weight values in $\boldsymbol{w}^{\text{final}}$ are $-\sigma$, 0, or $+\sigma$, with a ternary representation that can be efficiently deployed on hardware [23].

2.3.3 Privacy Considerations

Privacy is another challenge in FL as the model updates (in our case, $\boldsymbol{M}^{k,t}$s) may leak information about the client data. Specifically, Differential privacy (DP) guarantees that the probability of an outcome of an algorithm that runs on client data does not change much by a single client's data. This is typically ensured via injecting noise to a function of the client data at a particular step in the algorithm with some utility loss in the application. While there have been many DP strategies developed for FL and deep learning [15, 172, 18, 28], these strategies typically suffer from severe performance degradation due to noise injection. To make DP practical, researchers have explored certain randomization mechanisms that amplify the privacy guarantee. When these mechanisms are parts of the FL framework, such as sampling (data [35, 259] or device [36, 97, 107]) and shuffling [84, 87], the amplification comes for free. This is helpful because the overall process can meet a stronger privacy guarantee without increasing the noise level. `FedPM` promises one such amplification due to the stochastic Bernoulli sampling step. We first revisit the definitions of differential privacy [80], Rényi divergence, and Rényi differential privacy [173]; and then present the amplification result.

Definition 1. ***[Adjacent Datasets]*** *Two datasets $D, D' \in \mathcal{D}$ are called adjacent if they differ in at most one data sample.*

Definition 2. ***[(ϵ, δ)-DP]*** *A randomized mechanism $f : \mathcal{D} \to \mathcal{R}$ offers (ϵ, δ)-differential privacy if for any adjacent $D, D' \in \mathcal{D}$ and $\mathcal{S} \subset \mathcal{R}$*

$$Pr[f(D) \in \mathcal{S}] \leq e^{\epsilon} Pr[f(D' \in \mathcal{S})] + \delta.$$

Definition 3. ***[Rényi Divergence]*** *For two probability distributions P and Q defined over $\mathcal{R}$,*

the Rényi divergence of order $\alpha > 1$ is

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha.$$

Definition 4. ***[(α, ϵ)-RDP]*** *A randomized mechanism $f : \mathcal{D} \to \mathcal{R}$ offers ϵ-Rényi differential privacy of order α (or in short (α, ϵ)-RDP) if for any adjacent $D, D' \in \mathcal{D}$, it holds that*

$$D_\alpha(f(D)||f(D')) \leq \epsilon.$$

In particular, in [122] the authors have shown that when a sample $\boldsymbol{M} \in \{0,1\}^d$ from an already privatized vector $\boldsymbol{\theta} \in [c, 1-c]^d$, where $0 < c < 0.5$, is released to a third party (instead of $\boldsymbol{\theta}$ itself), the privacy is amplified under some conditions. More precisely, when there is an (α, ϵ)-Rényi Differential Privacy mechanism [173] that privatizes $\boldsymbol{\theta} \in [c, 1-c]^d$, releasing a sample from Bern$(\boldsymbol{\theta})$ yields an improved privacy budget (the smaller ϵ, the better the privacy): $\epsilon_{amp} \leq \min\{\epsilon, d \cdot r_\alpha(c)\}$. Here, $r_\alpha(p)$ is the binary symmetric Rényi divergence function defined as

$$r_\alpha(p) = \frac{1}{\alpha - 1} \log \left(p^\alpha (1-p)^{1-\alpha} + (1-p)^\alpha p^{1-\alpha} \right).$$

Notice that `FedPM` already involves this Bernoulli sampling step in the communication protocol and in the forward pass $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$. However, the d term in the upper bound limits the amplification for large model sizes. We believe it is worth exploring a tighter upper bound on ϵ_{amp} to enjoy privacy amplification in `FedPM` with practical models. Nonetheless we demonstrate the impact of this amplification on a distributed mean estimation problem, described in Figure 2.8, where the goal is to estimate the true mean of the probability masks $\bar{\boldsymbol{\theta}} = \frac{1}{K} \sum_{k=1}^K \boldsymbol{\theta}^k$ under communication and privacy constraints. We also provide a bias correction mechanism specific to our scheme in Figure 2.9, that mitigates the bias due to the DP mechanism and reduces the estimation error.

Now, suppose that we have an (α, ϵ)-RDP algorithm f that outputs privatized $\boldsymbol{\theta}^k \in [c, 1-c]^d$ with $0 < c < 0.5$, using local client data $\mathcal{D}_k$. As summarized in Figure 2.9, we are interested in what happens when instead of releasing $\boldsymbol{\theta}^k = f(\mathcal{D}_k)$, the client k releases a Bernoulli sample from it: $\boldsymbol{m}^k \in \{0,1\}^d \sim \text{Bern}(\boldsymbol{\theta}^k)$. We already explained the advantages in terms of communication bitrate, estimation error, unbiasedness throughout the manuscript; however, this approach also amplifies the privacy guarantees, meaning that it makes the overall privacy budget smaller $\epsilon_{amp} \leq \epsilon$. Quantitatively, [122] showed that after the Bernoulli sampling, the privacy budget of the overall process is

$$\epsilon_{amp} \leq \min\{\epsilon, d r_\alpha(c)\},$$

where $r_\alpha(\cdot)$ is the Rényi divergence of the binary symmetric function. More precisely, consider P, Q random variables with support on $\{x_1, x_2\} \subset \Theta$ and let $p = \Pr[P = x_1]$, $1 - p = \Pr(Q = x_1)$. Then the Rényi divergence is defined as

$$r_\alpha(p) = R_\alpha(P, Q) = \frac{1}{\alpha - 1} \log \left(p^\alpha (1-p)^{1-\alpha} + (1-p)^\alpha p^{1-\alpha}\right).$$

Notice that `FedPM` already involves this Bernoulli sampling step in the communication protocol and in the forward pass $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$. This implies that `FedPM` improves the privacy guarantee without changing the privacy mechanism – e.g. without increasing the injected noise level. However, the d term in the upper bound limits the amplification for large model sizes. We believe it is worth exploring a tighter upper bound on ϵ_{amp} to enjoy privacy amplification in `FedPM` with practical models. Nonetheless, we demonstrate the impact of this amplification on a distributed mean estimation problem, described in Figure 2.9, where the probability masks $\boldsymbol{\theta}^k \in [c, 1-c]^d$ are a function of client data $\mathcal{D}_k$; and are first corrupted by Gaussian noise, and then clipped to the range $[c, 1-c]^d$. Our goal is, as before, to estimate the true mean $\bar{\boldsymbol{\theta}} = \frac{1}{K} \sum_{k \in \mathcal{K}_t} \boldsymbol{\theta}^k$ by averaging the sampled binary masks, i.e., $\hat{\bar{\boldsymbol{\theta}}} = \frac{1}{K} \sum_{k \in \mathcal{K}_t} \boldsymbol{m}^k$. Differently from our previous experiments, we have privacy constraints now, meaning that we want to guarantee (ϵ, δ)-DP by injecting a Gaussian noise with variance $\sigma^2 = \frac{2 \ln (1.25/\delta) \Delta_2^2}{\epsilon^2}$ with a small ϵ, where $\delta \approx \frac{1}{N^2}$ and Δ_2 is the ℓ_2-sensitivity of the probability masks (in our case $\Delta_2 = (1-2c)\sqrt{d}$). We transfer the above amplification results in RDP to DP using the well-known relation:

Remark 1. *In [173] it is shown that if f is an (α, ϵ)-RDP mechanism, it also satisfies $(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta)$-DP for any $0 < \delta < 1$.*

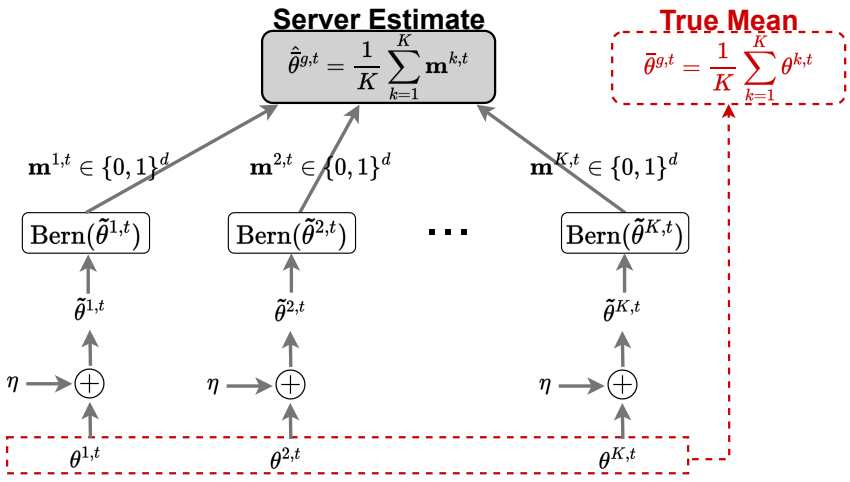


Figure 2.9: Distributed mean estimation scheme in `FedPM`, modified for differential privacy.

Since clipping after the noise addition step would lead to bias in the estimated mean, we work out a bias correction mechanism. We denote with θ one general parameter at client k for one parameter, with $\tilde{\theta}$ its noisy version, and with $\hat{\theta} = \text{clip}(\tilde{\theta})$ its clipped version. Specifically, if

$\tilde{\theta} = \theta + \eta$ is the noisy version of the parameter, where $\eta \sim \mathcal{N}(0, \sigma^2)$, then

$$\text{clip}(\tilde{\theta}) = \begin{cases} \tilde{\theta}, & \text{if} \quad c \leq \theta + \eta \leq 1 - c \\ 1 - c, & \text{if} \quad \theta + \eta > 1 - c \\ c, & \text{if} \quad \theta + \eta < c. \end{cases} \tag{2.18}$$

We now compute $\mathbb{E}\left[\hat{M}\right]$, where $\hat{M} \sim \text{Bern}(\hat{\theta})$, to analyze the bias $\mathbb{E}\left[\hat{M}\right] - \mathbb{E}[M] = \mathbb{E}\left[\hat{M}\right] - \theta$, where $M \sim \text{Bern}(\theta)$. First of all, notice that

$$\mathbb{E}\left[\hat{M}\right] = \int_0^1 \mathbb{E}\left[\hat{M}|\hat{\theta} = \rho\right] f(\rho)d\rho = \int_0^1 \rho f(\rho)d\rho = \mathbb{E}[\hat{\theta}].$$

And we now compute the mean of the clipped parameter

$$\begin{aligned}
\mathbb{E}\left[\hat{\theta}\right] &= \int_0^1 \rho f(\rho)d\rho \\
&= \int_{-\infty}^{+\infty} \text{clip}(\theta + \eta) f(\eta)d\eta \\
&= \int_{-\infty}^{c-\theta} c \cdot f(\eta)d\eta + \int_{c-\theta}^{1-c-\theta} (\theta + \eta) \cdot f(\eta)d\eta + \int_{1-c-\theta}^{+\infty} (1 - c) \cdot f(\eta)d\eta \\
&= c\Phi_\sigma(c - \theta) + \theta \int_{c-\theta}^{1-c-\theta} f(\eta)d\eta + \int_{c-\theta}^{1-c-\theta} \eta f(\eta)d\eta + (1 - c)\left(1 - \Phi_\sigma\left(1 - c - \theta\right)\right) \\
&= c\Phi_\sigma(c - \theta) + \theta\left[\Phi_\sigma(1 - c - \theta) - \Phi_\sigma(c - \theta)\right] + \frac{-\sigma}{\sqrt{2\pi}}\left[e^{\frac{-(1-c-\theta)^2}{2\sigma^2}} - e^{\frac{-(c-\theta)^2}{2\sigma^2}}\right] + \\
&\quad + (1 - c)\left(1 - \Phi_\sigma\left(1 - c - \theta\right)\right) \\
&= 1 - c + [\theta - 1 + c]\Phi_\sigma(1 - c - \theta) + [c - \theta]\Phi_\sigma(c - \theta) + \frac{-\sigma e^{\frac{-(c-\theta)^2}{2\sigma^2}}}{\sqrt{2\pi}}\left[e^{-2(c-\theta)-1} - 1\right],
\end{aligned}$$

where $\Phi_\sigma(\cdot)$ is the cumulative distribution function of a Gaussian random variable with standard deviation σ, and zero mean. We use this relation to correct the bias in $\hat{\bar{\boldsymbol{\theta}}}$. In practice, to adopt the bias-correction strategy, we sample the function $\mathbb{E}\left[\hat{\theta}\right]$, which is a function of the true parameter θ, noise standard deviation σ, and clipping parameter c, at Q different points $x_1, \ldots, x_Q$, i.e., different values for the uncorrupted θ, and we store the values in a table. Indeed, the values σ and c are set at the beginning of the training process, secretly shared among the participants, and never modified. Then, once the server computes an estimate for $\hat{\theta}$, it corrects it by finding the closest outputs of $\mathbb{E}\left[\hat{\theta}\right]$ in the stored table, and it inverts the map by choosing the corresponding x_i, i.e., the original θ.

To empirically assess the performance of the proposed solution, we conduct our experiments on a toy example with $N = 100$ clients, each having independent probability masks with dimension $d = 5$ and range $[0.2, 0.8]$, i.e., $\boldsymbol{\theta} \in [0.2, 0.8]^5$. Figure 2.10 shows the estimation error $||\hat{\bar{\boldsymbol{\theta}}}^{g,t} - \bar{\boldsymbol{\theta}}^{g,t}||_2^2$ under no noise injection case (i.e. no DP) with the black line. Recall that we want to reach

a smaller estimation error and smaller ϵ (i.e., a stronger privacy guarantee). The red curve corresponds to the ϵ vs. estimation error behavior if Bernoulli sampling did not amplify the privacy. The blue curve shows the amplified ϵ (i.e. $\epsilon_{amp} \leq \epsilon$) vs. estimation error behavior, and it overlaps with the red curve for ϵ values smaller than $d \cdot r_\alpha(c) = 8.96$, where there is no privacy amplification, i.e., $\epsilon_{amp} = \epsilon$. However, notice that the blue line never reaches ϵ's higher than this value due to amplification, while enjoying smaller estimation errors that the red curve can only achieve with very large ϵ. This shows the promise of `FedPM` in having a better privacy-accuracy performance than most baselines that do not have amplification. Finally, the green curve shows that bias correction improves this performance further even with $\epsilon < d \cdot r_\alpha(c) = 8.96$ by achieving lower estimation errors with the same ϵ.

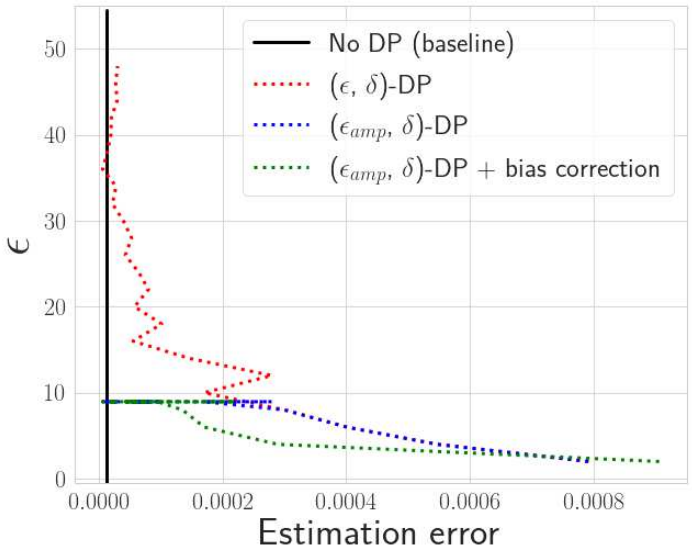


Figure 2.10: The effect of privacy amplification and bias correction in the privacy budget (ϵ) vs. estimation error behavior. Comparing red and blue curves, we see that we can reach small estimation errors without increasing ϵ thanks to the amplification (see the vertical blue line at low estimation error.). While the red curve and blue curve overlap for $\epsilon < d \cdot r_\alpha(c) = 8.96$, in that regime, we benefit from our bias correction strategy to reach a lower error.

2.3.4 Experiments

In this section, we empirically show the performance of `FedPM` in terms of accuracy, bitrate, converge speed, and the final model size. We consider four datasets: CIFAR-10 with 10 classes, CIFAR-100 [142] with 100 classes, MNIST [77] with 10 classes, and EMNIST [66] with 47 classes. For CIFAR-100, we use a 10-layer convolutional network (CNN) `CONV-10` and ResNet-18 [111]; for CIFAR-10, a 6-layer CNN `CONV-6` and ResNet-18 [111]; and for MNIST and EMNIST, a 4-layer CNN `CONV-4`. A detailed description of the architectures can be found in Tab. 2.1. We first

compare FedPM with SignSGD [44], TernGrad [262], QSGD [24], DRIVE [246], EDEN [247], and FedMask [153] on IID data split and full client participation. We then extend our experiments to non-IID data splits and partial participation. Finally, we present a key ablation study to justify why the Bayesian aggregation strategy is necessary for partial participation and to demonstrate how the resetting frequency affects the convergence rate and the final accuracy. In all the experiments, clients perform 3 local epochs with a batch size of 128 and a local learning rate of 0.1 in all the experiments. Notice that there is no server learning rate in FedPM; instead, we tune the prior resetting schedule in Bayesian aggregation in the case of non-IID data splits. We conducted our experiments on NVIDIA Titan X GPUs on an internal cluster server, using 1 GPU per one run. The code is publicly available. [3]

Table 2.1: Architectures for CONV-4, CONV-6, and CONV-10 models used in the experiments.

Model	CONV-4	CONV-6	CONV-10
Convolutional Layers	64, 64, pool 128, 128, pool	64, 64, pool 128, 128, pool 256, 256, pool	64, 64, pool 128, 128, pool 256, 256, pool 512, 512, pool 1024, 1024, pool
Fully-Connected Layers	256, 256, 10	256, 256, 10	256, 256, 100

IID Data Split and Full Participation ($K = N$)

In this section, we focus on IID data distribution and the case when all the clients participate in the training at each round. We set the number of clients to $N = K = 10$. We report the estimated bitrate for the arithmetic code that uses the empirical frequency of the symbols (for our method FedPM, this corresponds to the frequency of 1's in $\boldsymbol{m}^{k,t}$) – which is equal to the empirical entropy for blocklength d as large as the model size. In Figure 2.11, we compare the accuracy, bitrate, and convergence speed of FedPM with relevant baselines. As can be seen in the figure, FedPM converges to the highest accuracy on all four datasets. DRIVE, EDEN, and QSGD (they mostly overlap in the accuracy plots) seem to be the three baselines that perform the best after FedPM; however, their convergence speed is significantly lower than FedPM. In terms of convergence speed, FedMask is the fastest among the baselines – in fact, at the beginning of the training, FedMask is faster than FedPM as well. However, its final accuracy is lower than the others. We also would like to highlight that while some of our baselines, such as FedMask and TernGrad, have a visibly high variance in accuracy, FedPM shows stable training behavior across all experiments.

In terms of bitrate, SignSGD and FedMask consistently spend 1 bpp, which is the default number when a binary mask or sign mask is communicated. This means binary values (1's and 0's) are almost equally distributed in their masks, which prevents them from enjoying additional

[3] https://github.com/BerivanIsik/sparse-random-networks

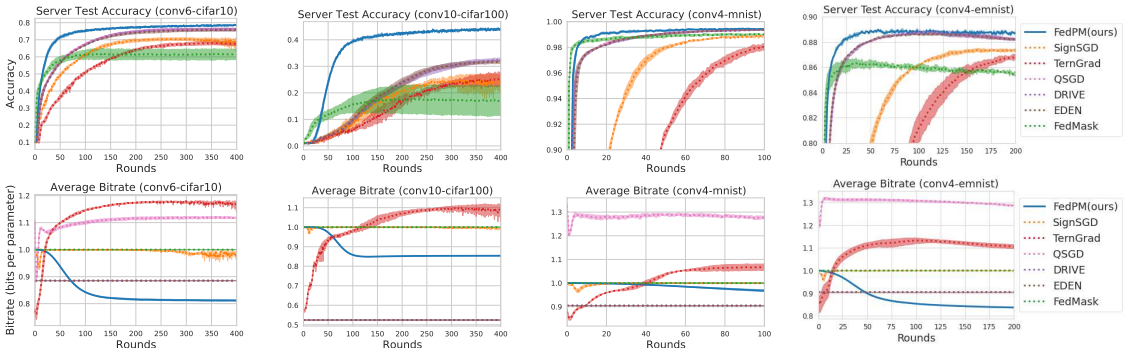


Figure 2.11: Accuracy and bitrate comparison of `FedPM` with SignSGD [44], TernGrad [262], QSGD [24], DRIVE [246], EDEN [247], and FedMask [153], all performing in the same bitrate regime.

bitrate gains. Across all experiments, TernGrad has the highest bitrate. We would like to leave a note about the bitrate of QSGD. Unlike other baselines, including our work, QSGD can go down to very low bitrates by adjusting the number of levels in quantization. We have observed that in the extreme quantization case, QSGD underperforms `FedPM`. Then, we have decided to increase the number of quantization levels in QSGD to see if it improves the accuracy. However, as can be seen from the plots, even with bitrate larger than 1, QSGD still underperforms `FedPM`. The only two baselines that challenge `FedPM` in terms of bitrate are DRIVE and EDEN. While `FedPM` has lower bitrates on CIFAR-10 and EMNIST; DRIVE and EDEN have better bitrates on CIFAR-100 and MNIST. However, the accuracy of DRIVE and EDEN on these datasets (specifically CIFAR-100) is significantly lower than that of `FedPM`, with slower convergence.

As for the final model size, `FedPM` needs only 0.8 bpp for the `CONV-6` model trained on CIFAR-10, 0.85 bpp for the `CONV-10` model trained on CIFAR-100, 0.96 bpp for the `CONV-4` model trained on MNIST, and 0.83 bpp for the `CONV-4` model trained on EMNIST. On the other hand, other baselines that train a dense model, namely SignSGD, TernGrad, QSGD, DRIVE, and EDEN, would need to represent each weight with their full precision value, i.e., 32 bpp. This implies that `FedPM` provides around 38.6× improvement in the storage or the communication of the final model. Since FedMask also trains a sparse model, it enjoys a similar gain in the final model size requiring 1 bpp across all the models. Due to the stochastic masking procedure and uneven distribution of 1's and 0's in the binary masks, `FedPM` has up to 0.17 bpp improvement over the deterministic procedure in FedMask, which adds up to a large gain due to the huge model size.

We provide additional experimental results with ResNet-18 model on CIFAR-10 and CIFAR-100 datasets in Appendix 2.3.4; and observe similar improvements over the baselines.

Non-IID Data Split and Partial Participation ($K < N$)

This section considers more realistic scenarios, in which the local clients' datasets are generated from slightly different data distributions. We focus on CIFAR-10 with `CONV-6`, and we compare `FedPM` against (i) the most promising baselines, which, based on the previous, are DRIVE,

	Algorithm	$\rho = 1$	$\rho = 0.5$	$\rho = 0.2$	$\rho = 0.1$
$c_{\max} = 4$	DRIVE [246]	0.739 ± 0.005	0.632 ± 0.010	0.563 ± 0.005	0.405 ± 0.018
	EDEN [247]	0.717 ± 0.006	0.665 ± 0.012	0.565 ± 0.009	0.360 ± 0.016
	QSGD [24]	0.709 ± 0.006	0.644 ± 0.014	0.567 ± 0.010	0.399 ± 0.020
	FedMask [153]	0.531 ± 0.044	0.435 ± 0.057	0.434 ± 0.036	0.362 ± 0.024
	`FedPM` (Ours)	**0.748 ± 0.003**	**0.720 ± 0.007**	**0.617 ± 0.021**	**0.496 ± 0.007**
$c_{\max} = 2$	DRIVE [246]	0.434 ± 0.025	0.376 ± 0.014	**0.375 ± 0.015**	0.221 ± 0.003
	EDEN [247]	0.535 ± 0.050	0.461 ± 0.016	**0.380 ± 0.015**	0.219 ± 0.005
	QSGD [24]	0.476 ± 0.033	0.464 ± 0.002	**0.375 ± 0.026**	0.243 ± 0.014
	FedMask [153]	0.420 ± 0.028	0.387 ± 0.062	0.285 ± 0.040	0.197 ± 0.030
	`FedPM` (Ours)	**0.643 ± 0.016**	**0.556 ± 0.031**	**0.372 ± 0.004**	**0.277 ± 0.003**

Table 2.2: Average final accuracy $\pm\sigma$ in non-IID data split with $c_{\max} = 4$ and 2, and client participation ratios $\rho = \{0.1, 0.2, 0.5, 1\}$, for `FedPM`, FedMask, and the strongest baselines in the IID experiments: EDEN, DRIVE, and QSGD. The training duration was set to $t_{\max} = 200$ rounds.

EDEN, and QSGD, and (ii) FedMask, as it is the only *sparse* baseline. To choose the size of each dataset $|\mathcal{D}_n| = D_n$, for each client $n \in \{1, \dots, N\}$, an integer j_n is sampled uniformly from $\{10, 11, \dots, 100\}$. Then, a coefficient $p_n = \frac{j_n}{\sum_n j_j}$ is computed, which represents the size of the local dataset D_n as a fraction of the size of the full dataset, i.e., the training set of CIFAR-10. In this way, highly unbalanced datasets can be generated from the central one. Moreover, since the task is a classification problem, we impose a maximum number of different labels, or classes, $c_{\max}$, that one client can see. Consequently, clients need cooperation to learn the statistics of other classes' distributions, as the test dataset contains samples from all classes. In addition, partial participation is also considered, meaning that at each round, the server uniformly samples a fraction $\rho = \frac{K}{N}$ of the clients to participate in the training round. This is motivated in real-world scenarios by the scarcity of physical communication network resources, which may limit the availability of part of the clients during one round. The maximum number of classes per local dataset is set to $c_{\max} \in \{2, 4\}$, and the participation ratio is set to $\rho \in \{0.1, 0.2, 0.5, 1\}$. For $\rho = 1$ and $\rho = 0.5$, the total number of clients is set to $N = 10$ (and so K is equal to 10 and 5, respectively). For $\rho = 0.2$, we set $N = 100$ (and so $K = 20$), and for $\rho = 0.1$, we set $N = 50$ (and so $K = 5$), which is the worst scenario among all combinations, given the small amount of information the server can collect at the end of each round. When $\rho = 1$, for the `FedPM` algorithm, we keep the same aggregation strategy exposed in Figure 2.8; and we switch to the Bayesian aggregation method (see Section 2.3.2) when there is partial participation, i.e., when $\rho < 1$. Indeed, applying the Bayesian aggregation method is revealed to be crucial for achieving good accuracy when $\rho < 1$ and data are non-IID, obtaining a large gain with respect to the simpler version in Figure 2.8, which resets the Beta priors at each round (or takes the average of the samples, as explained in Section 2.3.2). We elaborate more on this observation with an ablation study in the next section. We adopt a simple heuristic schedule to reset the priors: Reset every 3 rounds when $\rho = 0.5$ and $\rho = 0.2$, and every 10 rounds when $\rho = 0.1$. As expected, the smaller the ratio ρ, the larger the number of rounds we should wait before resetting the priors to collect more information from a much more diverse

	Algorithm	$\rho = 1$	$\rho = 0.5$	$\rho = 0.2$	$\rho = 0.1$
$c_{\max} = 4$	DRIVE [246]	$0.885 \pm 9 \cdot 10^{-5}$	$\mathbf{0.885 \pm 1 \cdot 10^{-4}}$	$\mathbf{0.885 \pm 6 \cdot 10^{-5}}$	$\mathbf{0.885 \pm 1 \cdot 10^{-4}}$
	EDEN [247]	$0.885 \pm 1 \cdot 10^{-4}$	$\mathbf{0.885 \pm 1 \cdot 10^{-4}}$	$\mathbf{0.885 \pm 8 \cdot 10^{-5}}$	$\mathbf{0.885 \pm 1 \cdot 10^{-4}}$
	QSGD [24]	0.982 ± 0.027	0.923 ± 0.029	1.188 ± 0.034	0.910 ± 0.05
	FedMask [153]	$1.000 \pm 3 \cdot 10^{-6}$	$1.000 \pm 8 \cdot 10^{-8}$	$1.000 \pm 2 \cdot 10^{-6}$	$1.000 \pm 6 \cdot 10^{-7}$
	`FedPM` (Ours)	$\mathbf{0.863 \pm 0.077}$	0.912 ± 0.056	$0.965 \pm 1 \cdot 0.01812$	0.996 ± 0.003
$c_{\max} = 2$	DRIVE [246]	$0.885 \pm 7 \cdot 10^{-5}$	$\mathbf{0.885 \pm 2 \cdot 10^{-4}}$	$\mathbf{0.885 \pm 7 \cdot 10^{-5}}$	$\mathbf{0.885 \pm 2 \cdot 10^{-4}}$
	EDEN [247]	$0.885 \pm 1 \cdot 10^{-4}$	$\mathbf{0.885 \pm 7 \cdot 10^{-5}}$	$\mathbf{0.885 \pm 6 \cdot 10^{-5}}$	$\mathbf{0.885 \pm 7 \cdot 10^{-5}}$
	QSGD [24]	1.230 ± 0.043	1.234 ± 0.038	1.100 ± 0.01	1.082 ± 0.01
	FedMask [153]	$1.000 \pm 2 \cdot 10^{-6}$	$1.000 \pm 2 \cdot 10^{-6}$	$1.000 \pm 1 \cdot 10^{-5}$	$1.000 \pm 2 \cdot 10^{-7}$
	`FedPM` (Ours)	$\mathbf{0.868 \pm 0.076}$	0.904 ± 0.063	0.980 ± 0.014	0.997 ± 0.01

Table 2.3: Average bitrate $\pm\sigma$ over the whole training process in non-IID data split with $c_{\max} = 4$ and $c_{\max} = 2$, and partial participation with ratios $\rho = \{0.1, 0.5, 1\}$, for `FedPM`, FedMask, and the strongest baselines in the IID experiments: EDEN, DRIVE, and QSGD. The training duration was set to $t_{\max} = 200$ rounds.

pool of clients.

Table 2.2 reports the results with $c_{\max} = 4$ and 2. `FedPM` seems to outperform all the baselines in every configuration, as the Bayesian aggregation allows the server to collect more data before resetting the priors, which is important when clients' data distributions are non-IID, and only a fraction of the clients participate in each round. This strategy can be seen as the `FedPM` counterpart of decreasing the learning rate (which we applied in the other *dense* compression-based baselines, like DRIVE, EDEN, and QSGD). It is seen from Table 2.2 that FedMask [153] is struggling in the non-IID case, as applying a hard threshold on the scores to binarize the mask does not provide a proper way to implement multiple-rounds aggregation, emphasizing the benefit of the stochastic process in `FedPM`. It is interesting to notice that, especially when $c_{\max} = 4$, the lower the value of ρ, the larger the gap between `FedPM` and the baselines, corroborating the fact that the Bayesian strategy can better deal with partial participation.

We now express some communication bitrate considerations on such experiments. Table 2.3 reports the average bitrate needed by different algorithms over the whole training process when $c_{\max} = 4$ and $c_{\max} = 2$. By simply multiplying the obtained average bitrate by the total number of rounds $t_{\max} = 200$, we obtain the total number of bits one element in the global probability mask needs to converge to its final value, indicating the total amount of information communicated during the training process.

We first observe that both DRIVE and EDEN consume almost the same amount of bits no matter the system configuration and round number (very small variance), and it is instead model dependent (see Figure 2.11). On the contrary, `FedPM` and QSGD report higher bitrate variability, as it depends on both the training phase and system setting. As already observed in Figure 2.11, FedMask balances almost uniformly the binary updates, leading to a bitrate that is basically fixed to 1. For both $c_{\max} = 4$ and $c_{\max} = 2$, `FedPM` yields the smallest bitrate when $\rho = 1$, whereas for the other scenarios, EDEN and DRIVE are slightly more efficient. We argue that this is motivated by the fact that, as the learning task becomes harder due to the high system heterogeneity, all the

models struggle to converge to good and stable solutions, which means that FedPM is still uncertain about the *weights' importance probabilities* $\boldsymbol{\theta}$, setting many of them close to 0.5. However, we think that this may be a useful feature of FedPM to quantify its internal uncertainty, which we will further analyze.

To conclude the analysis, we also report the FedPM bpp for the final model, which is an indication of the average number of bits needed per one parameter of the model. In the case of $c_{\max} = 4$, the final model sizes are 0.79 bpp, 0.834 bpp, and 0.99 bpp, when $\rho = \{0.1, 0.5, 1\}$, respectively. When $c_{\max} = 2$, the final model sizes are 0.8 bpp, 0.817 bpp, and 0.992 bpp. Consequently, at the end of the training process, FedPM remains the most efficient option.

Ablation Study on the Bayesian Aggregation Strategy

In this section, we try to answer two questions: (1) *Is Bayesian aggregation really necessary?* and (2) *What is the effect of resetting frequency on the convergence rate and the final accuracy?* We do this by analyzing the effect of different resetting frequencies of the Beta priors on the training behavior of FedPM with non-IID data split and partial client participation; and report the results in Figure 2.12. Hereafter, we denote with γ the number of aggregation rounds before resetting the priors. For instance, $\gamma = 1$ corresponds to resetting the priors at every iteration, which is equivalent to the aggregation method presented in Figure 2.8. On the other extreme, $\gamma = 200$ indicates that the priors are never reset. It is seen that $\gamma = 1$ curves fluctuate significantly and never converge to the best accuracy in any setting, while $\gamma = 200$ curves look smoother but converge to the lowest accuracy in all settings. This intuitively makes sense because, as already mentioned in Section 2.3.2, by increasing the value of γ, we allow the server to consider the information coming from multiple rounds while updating the global parameters. Indeed, with partial participation and non-IID data, a single round's updates may convey skewed information, depending on the level of data heterogeneity $c_{\max}$, and client participation ratio ρ. As a rule of thumb for the resetting frequency value, we suggest tuning γ around the value $\frac{1}{\rho}$. The rationale behind this is that with uniform client sampling, at least $\frac{1}{\rho}$ rounds are needed to have the non-zero probability to sample from each client once before resetting the prior. In practice, we do not need to sample exactly from every client, as enough information is contained in the updates of the other sampled ones.

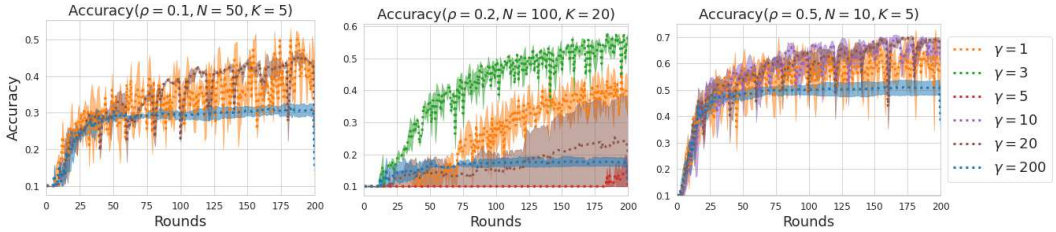


Figure 2.12: Accuracy for different values of γ – the number of rounds before resetting the priors.

In the non-IID and partial participation experiments to distill the final model we may apply

both stochastic sampling, as during training, or a hard-threshold method, similar to the one adopted in FedMask [153]. In the latter, a binary mask coefficient m_i is set to 1 if $\theta_i > \alpha_{\text{ths}}$, and 0 otherwise. For all experiments but one, when $\alpha_{\text{ths}} \in [0.4, 0.6]$, the thresholding test accuracy is always higher than the sampling method, and so we use the threshold method. However, in the extreme case $c_{\max} = 2$ and $\rho = 0.1$, the optimal values for $\alpha_{\max}$ were in $[0.2, 0.4]$ and $[0.6, 0.8]$ in all experiments, probably due to the high randomness given by the highly heterogeneous scenario. Consequently, for the last experiment, we just adopt the stochastic sampling strategy to evaluate the model, as further optimizing the α_{ths} means adapting to the test dataset, which may corrupt the ability of the model to generalize.

Additional Experiments on ResNet Architectures

In this section, we provide additional experimental results with ResNet-18 [111] on CIFAR-10 and CIFAR-100 datasets. For these experiments, we focus on IID data distribution and the case when all the clients participate in the training at each round. We provide the details of the ResNet-18 architecture in Table 2.4 below.

Table 2.4: ResNet-18 architecture.

Name	**Component**
conv1	3×3 conv, 64 filters. stride 1, BatchNorm
Residual Block 1	$\begin{bmatrix} 3 \times 3 \text{ conv, } 64 \text{ filters} \\ 3 \times 3 \text{ conv, } 64 \text{ filters} \end{bmatrix} \times 2$
Residual Block 2	$\begin{bmatrix} 3 \times 3 \text{ conv, } 128 \text{ filters} \\ 3 \times 3 \text{ conv, } 128 \text{ filters} \end{bmatrix} \times 2$
Residual Block 3	$\begin{bmatrix} 3 \times 3 \text{ conv, } 256 \text{ filters} \\ 3 \times 3 \text{ conv, } 256 \text{ filters} \end{bmatrix} \times 2$
Residual Block 4	$\begin{bmatrix} 3 \times 3 \text{ conv, } 512 \text{ filters} \\ 3 \times 3 \text{ conv, } 512 \text{ filters} \end{bmatrix} \times 2$
Output Layer	4×4 average pool stride 1, fully-connected, softmax

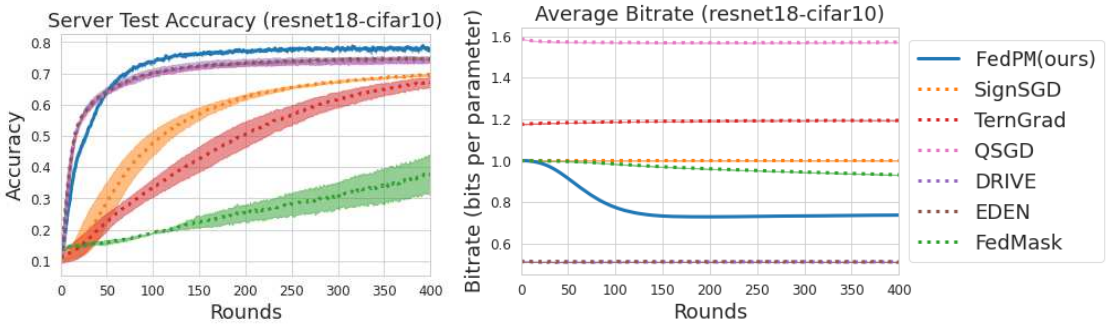


Figure 2.13: Accuracy and bitrate comparison of `FedPM` with baselines SignSGD [44], TernGrad [262], QSGD [24], DRIVE [246], EDEN [247], and FedMask [153], with ResNet-18 on CIFAR-10.

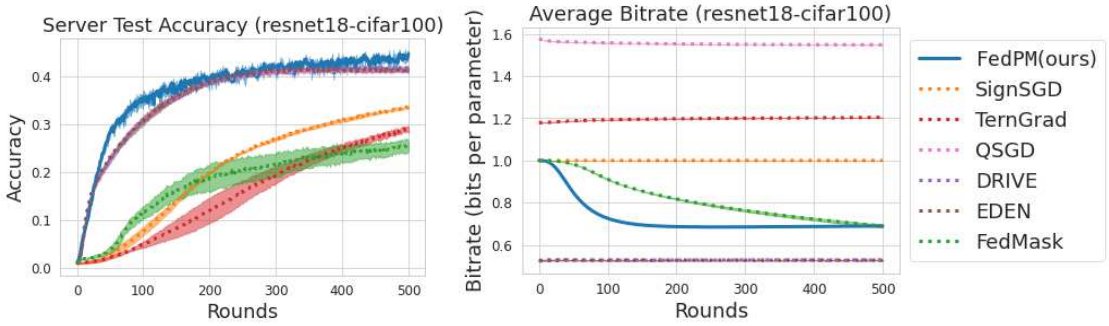


Figure 2.14: Accuracy and bitrate comparison of `FedPM` with baselines SignSGD [44], TernGrad [262], QSGD [24], DRIVE [246], EDEN [247], and FedMask [153], with ResNet-18 on CIFAR-100.

Figures 2.13 and 2.14 show the results on CIFAR-10 and CIFAR-100 datasets, respectively. It is seen that `FedPM` outperforms all the baselines in terms of accuracy. Although DRIVE and EDEN require approximately 0.1 smaller bitrates than `FedPM`, they also reach lower accuracy. In summary, the advantages of `FedPM` discussed in the main manuscript carry over to ResNet-18 model as well.

2.3.5 Conclusion

In this work, we introduced Federated Probabilistic Mask Training (`FedPM`) – a communication-efficient FL strategy. `FedPM` relies on the idea of finding a sparse network in a randomly initialized dense network, which is then sparsified by a collaboratively trained *stochastic* binary mask. In addition to reducing the communication cost to less than 1 bit per parameter (bpp), `FedPM` also reaches higher accuracy with faster convergence than the relevant baselines, and can potentially amplify privacy while additionally outputting a compressed final model with a size less than 1 bpp.

2.4 Communication-Efficient Federated Learning through Importance Sampling

2.4.1 Introduction

In Section 2.1 and Section 2.3 we discussed some techniques, mainly based on model compression, to reduce the communication burden in FL. However, many of these strategies adopt a stochastic approach that requires the client n to send a sample $\mathbf{x}^{(t,n)}$ from a client-only distribution $q_{\phi^{(t,n)}}$ (which we call the *post-data distribution*), while the goal of the server is to estimate $\mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N}X^{(t,n)}\right]$ by taking the average of the samples across clients $\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(t,n)}$. Here, we denote by N the number of clients, by $[N]$ the set $\{1,\dots,N\}$, and by

$\mathbf{a}_i^{(t,n)}$ the i-th parameter of a vector $\mathbf{a}$ at client n in round t. We show that in many stochastic FL settings, the server also holds a distribution $p_{\theta^{(t)}}$ (which we call the *pre-data distribution*) that is close to the post-data distribution $q_{\phi^{(t,n)}}$ (which is unknown to the server) in KL divergence. The proposed method, KL Minimization with Side Information (`KLMS`), exploits this closeness to reduce the cost of communicating samples $\mathbf{x}^{(t,n)}$[191]. We briefly summarize three of such stochastic FL frameworks by pointing to the corresponding pre-data $p_{\theta^{(t)}}$ and post-data $q_{\phi^{(t,n)}}$ distributions as examples of three different setups: (i) learning probability distributions over subnetworks (or masks), (ii) learning deterministic model parameters using stochastic compressors, and (iii) learning probability distributions over model parameters; and provide a more exhaustive summary in Section 2.4.1 and Section 2.7.1. However, before that, we first give a rough outline of how `KLMS` actually works.

Before describing the details of our proposal in Section 2.4.2, we now briefly provide the key idea `KLMS` relies on: Instead of communicating the deterministic value of a sample $\mathbf{x}^{(t,n)} \sim q_{\phi^{(t,n)}}$, client n can communicate a sample $\mathbf{y}^{(t,n)}$ from another distribution $\mathbf{y}^{(t,n)} \sim \tilde{q}_{\pi^{(t,n)}}$, which is less costly to communicate compared to $\mathbf{x}^{(t,n)}$, and the discrepancy due to sampling from this distribution is not significant. As shown in Algorithm 2.1, to construct $\tilde{q}_\pi$, we use the pre-data distribution $p_{\theta^{(t)}}$ (which is known by the server and the clients) and the importance sampling algorithm in [58].

Algorithm 2.1 `KLMS` Outline. (A more detailed description is given in Section 2.4.2 and Algorithm 2.7.1.)

---

(1) The server and client n generate the *same* K samples from the pre-data distribution $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^K \sim p_{\theta^{(t)}}$ (which is available to both the server and the clients) using a shared random seed.

(2) Client n computes the importance weights $\alpha_{[k]} = \frac{q_{\phi^{(t,n)}}(\mathbf{y}_{[k]}^{(t,n)})}{p_{\theta^{(t)}}(\mathbf{y}_{[k]}^{(t,n)})}$ for $k \in [K]$ with the local post-data distribution $q_{\phi^{(t,n)}}$ and normalizes it to get a distribution over $[K]$ as $\pi^{(t,n)}(k) = \frac{\alpha_{[k]}}{\sum_{l=1}^K \alpha_{[l]}}$.

(3) Client n takes a sample from this new distribution $k^{(n)*} \sim \pi^{(t,n)}$ and sends it to the server in $\log K$ bits.

(4) The server receives $k^{(n)*}$ and recovers the $k^{(n)*}$-th sample $\mathbf{y}_{[k^{(n)*}]}^{(t,n)}$ from the set of K samples $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^K$ generated from $p_{\theta^{(t)}}$ in Step (1). Notice that $\mathbf{y}_{[k^{(n)*}]}^{(t,n)}$ is actually a sample from the underlying distribution over $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^K$ defined as $\tilde{q}_{\pi^{(t,n)}}(\mathbf{y}) = \sum_{k=1}^K \pi^{(t,n)}(k) \cdot \mathbf{1}(\mathbf{y}_{[k]}^{(t,n)} = \mathbf{y})$.

We show that this procedure yields an arbitrarily small discrepancy in the estimation when $K \simeq \exp\left(D_{KL}(q_{\phi^{(n)}} \| p_\theta)\right)$ with theoretical and algorithmic improvements (specific to the FL setting) over prior work [108, 240]. Clearly, to get the most communication gain out of `KLMS`, we need pre-data p_θ and post-data $q_{\phi^{(n)}}$ distributions that are close in KL divergence. We show the existence of such distributions in many stochastic FL frameworks by providing concrete examples in the next section.

Setups

We now briefly summarize three examples of stochastic FL frameworks that KLMS can be integrated into by highlighting the natural choices for pre-data p_θ and post-data $q_{\phi^{(n)}}$ distributions.

FedPM [200] freezes the parameters of a randomly initialized network and finds a subnetwork inside it that performs well with the initial random parameters. To find the subnetwork, the clients receive a global probability mask $\theta^{(t)} \in [0,1]^d$ from the server that determines, for each parameter, the probability of retaining it in the subnetwork; set this as their local probability mask $\phi^{(t,n)} \leftarrow \theta^{(t)}$; and train only this mask (not the frozen random parameters) during local training. At inference, a sample $x^{(t,n)} \in \{0,1\}^d$ from the Bernoulli distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ is taken, and multiplied element-wise with the frozen parameters of the network, obtaining a pruned random subnetwork, which is then used to compute the model outputs. Communication consists of three stages: (i) clients update their local probability masks $\phi^{(t,n)}$ through local training; (ii) at the end of local training, they send a sample $x^{(t,n)} \sim \text{Bern}(\cdot;\phi^{(t,n)})$ to the server; (iii) the server aggregates the samples $\frac{1}{N}\sum_{n=1}^{N} x^{(t,n)}$, updates the global probability mask $\theta^{(t+1)}$, and broadcasts the new mask to the clients for the next round. FedPM achieves state-of-the-art results in accuracy-bitrate tradeoff with around 1 bit per parameter (bpp). As the model converges, the global probability mask $\theta^{(t)}$ and clients' local probability masks $\phi^{(t,n)}$ get closer to each other (see Figures 2.15 and 2.19 for the trend of $D_{KL}(q_{\phi^{(t,n)}}||p_{\theta^{(t)}})$ over time). However, no matter how close they are, FedPM employs approximately the same bitrate for communicating a sample from $\text{Bern}(\cdot;\phi^{(t,n)})$ to the server that knows $p_{\theta^{(t)}}$. We show that this strategy is suboptimal and applying KLMS with the global probability distribution $\text{Bern}(\cdot;\theta^{(t)})$ as the pre-data distribution $p_{\theta^{(t)}}$, and the local probability distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ as the post-data distribution $q_{\phi^{(t,n)}}$, provides up to 50 times gain in compression.

QSGD [24], different from the stochastic approach taken by FedPM to train a probabilistic mask, is proposed to train a deterministic set of parameters. However, QSGD is itself a stochastic quantization operation. More concretely, QSGD quantizes each coordinate $\mathbf{v}_i^{(t,n)}$ using the following probability distribution (which we call the QSGD distribution $p_{\texttt{QSGD}}(\cdot)$), where s is the number of quantization levels:

$$p_{\texttt{QSGD}}\left(\hat{\mathbf{v}}_i^{(t,n)}\right) = \begin{cases} \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} - \left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \frac{\|\mathbf{v}^{(t,n)}\|\cdot\text{sign}(\mathbf{v}_i^{(t,n)})}{s}\left(\left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor + 1\right) \\ 1 - \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} + \left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \frac{\|\mathbf{v}^{(t,n)}\|\cdot\text{sign}(\mathbf{v}_i^{(t,n)})}{s}\left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor \end{cases}. \tag{2.19}$$

QSGD takes advantage of the empirical distribution of the quantized values (large quantized values are less frequent) by using Elias coding to encode them – which is the preferred code when the small values to encode are much more frequent than the larger values [83]. However, QSGD still does not fully capture the distribution of the quantized values since Elias coding is not adaptive to the data. We fix this mismatch by applying KLMS with the QSGD distribution $p_{\texttt{QSGD}}(\cdot)$ as the post-data distribution $q_{\phi^{(t,n)}}$, and the empirical distribution induced by the historical updates at

the server from the previous round as the pre-data distribution $p_{\theta^{(t)}}$. These two distributions are expected to be *close* to each other due to the temporal correlation across rounds, as previously reported in [125, 184]. We demonstrate that KLMS exploits this closeness and outperforms vanilla QSGD with a 12 times improvement in bitrate.

Federated SGLD [82] targets a Bayesian FL setup, where the goal is to learn a global posterior distribution p_θ over the model parameters from clients' local posteriors $q_{\phi^{(n)}}$. A state-of-the-art method proposed in [253] is the federated counterpart of the Stochastic Gradient Langevin Dynamics (SGLD) [261], which uses a novel Markov Chain Monte Carlo (MCMC) algorithm. In this setting, the global posterior distribution is assumed to be proportional to the product $p_{\theta^{(t)}} \sim \prod_{n=1}^{N} e^{-U(\phi^{(t,n)})}$ of N local unnormalized posteriors associated with each client, expressed as potential functions $\{U(\phi^{(t,n)})\}_{n=1}^{N}$. At the beginning of each local training round, the local clients' posteriors are initialized with the global posterior $\phi^{(n,t)} \leftarrow \theta^{(t)},\ \forall n \in [N]$. Then the clients compute an unbiased estimate of their gradients $H(\phi^{(t,n)}) = \frac{|D^{(n)}|}{|\mathcal{S}^{(t,n)}|} \sum_{j \in \mathcal{S}^{(t,n)}} \nabla U_j(\phi^{(t,n)})$, where $|D^{(n)}|$ is the size of the local dataset of client n, and $\mathcal{S}^{(t,n)}$ is the batch of data used to estimate the gradient. They then communicate these estimates to the server, which aggregates them by computing

$$\theta^{(t+1)} = \theta^{(t)} - \gamma \sum_{n=1}^{N} H(\phi^{(t,n)}) + \sqrt{2\gamma}\xi^{(t)}, \tag{2.20}$$

where $\xi^{(t)}$ is a sequence of i.i.d. standard Gaussian random variables. As reported in [82, 253], the sequence of global updates $\theta^{(t)}$ converges to the posterior sampling. Notice that the clients communicate their gradient vectors $H(\phi^{(t,n)})$ to the server at every round, which is as large as the model itself. To reduce this communication cost, in [253], the authors propose a compression algorithm called QLSD that stochastically quantizes the updates with essentially the Bayesian counterpart of QSGD [24].

However, neither QLSD nor the other compression baselines in the Bayesian FL literature [59, 81, 203] take full advantage of the stochastic formulation of the Bayesian framework, where the server and the clients share side information (the global posterior $p_{\theta^{(t)}}$) that could be used to improve the compression gains. Instead, they quantize the updates ignoring this side information. This approach is suboptimal since (i) the precision is already degraded in the quantization step, and (ii) the compression step does not account for the side information $p_{\theta^{(t)}}$. We show that we can exploit this inherent stochastic formulation of Bayesian FL by applying KLMS with the global posterior distribution as the pre-data distribution $p_{\theta^{(t)}}$, and the local posterior distribution as the post-data distribution $q_{\phi^{(t,n)}}$. In addition to benefiting from the side information, KLMS does not restrict the message domain to be discrete (as opposed to the baselines) and can reduce the communication cost by 4 times, while also achieving higher accuracy than the baselines.

Contributions

We have listed three examples of stochastic communication-efficient FL frameworks, each of which induces a post-data distribution $q_{\theta^{(t,n)}}$ that clients want to send a sample from, and a pre-data distribution $p_{\theta^{(t)}}$ that is available to both the clients and the server – playing the role of side information. In each case, these distributions are expected to become closer in KL divergence as training progresses due to the convergence of the model parameters (`FedPM` or other probabilistic mask learning methods), temporal correlation across rounds (`QSGD` or other deterministic model training methods), or the stochastic formulation of the framework itself (Federated `SGLD` or other Bayesian FL methods). We show that `KLMS` reduces the communication cost down to this *fundamental quantity* (KL divergence) in each scenario, resulting in up to 50 times improvement in communication efficiency (sometimes with higher accuracies) over `FedPM`, `QLSD`, and `QSGD` among other non-stochastic competitive baselines such as `SignSGD` [44], `TernGrad` [262], `DRIVE` [246], `EDEN` [247], and `FedMask` [153]. To achieve this efficiency, we use an importance sampling algorithm [58, 106] by improving and extending the previous theoretical guarantees to the distributed setting. Different from prior work that used importance sampling in the centralized setting to compress model parameters [108] or focused on differential privacy implications [219, 240], `KLMS` selects more natural pre-data $p_{\theta^{(t)}}$ and post-data $q_{\phi^{(t,n)}}$ distributions that are intrinsic to the FL setting, and optimizes the bit allocation across both the training rounds and the model coordinates in an adaptive way to achieve the optimal bitrate, while also eliminating a hyperparameter required by prior work [108, 240]. Our contributions can be summarized as follows:

(1) We propose a road map to utilize various forms of side information available to both the server and the clients to reduce the communication cost in FL. We give concrete examples of how to code model updates under different setups, including probabilistic mask training (e.g., `FedPM`), deterministic model training with stochastic compressors (e.g., `QSGD`), and Bayesian FL (e.g., Federated `SGLD`).

(2) We extend the importance sampling results to the distributed setting with theoretical improvements.

(3) We propose an adaptive bit allocation strategy that eliminates a hyperparameter required by prior work, and allows a better use of the communication budget across the model coordinates and rounds.

(4) We demonstrate the efficacy of our strategy on MNIST, EMNIST, CIFAR-10, and CIFAR-100 datasets, and show improvements in accuracy with up to 50 times gains in bitrate (with sometimes higher accuracies) over relevant baselines such as `FedPM` [200], `QLSD` [253], `QSGD` [24], `SignSGD` [44], `TernGrad` [262], `DRIVE` [246], `EDEN` [247], `FedMask` [153], and `DP-REC` [240].

2.4.2 KL Divergence Minimization with Side Information (`KLMS`)

We first describe our approach, `KLMS`, together with theoretical guarantees; then, we introduce our adaptive bit allocation strategy to optimize the bitrate across training rounds and model coordinates to reduce the compression rate; finally,we give four concrete examples where `KLMS` improves the accuracy-bitrate tradeoff.

KLMS for Stochastic FL Frameworks

We first point out that our proposal is not a stand-alone FL framework to replace existing alternatives, rather, it represents a general recipe that can be integrated into many existing (stochastic) frameworks to improve their accuracy-bitrate performance significantly. The main idea behind KLMS is grounded in three important observations:

(1) In many existing FL frameworks, the updates communicated from the clients to the server are samples drawn from some optimized post-data distributions, e.g., QSGD [24] and FedPM [200].

(2) Sending a *random* sample from a distribution can be done much more efficiently than first taking a sample from the same distribution, and then sending its *deterministic* value [236].

(3) The knowledge acquired from the historical updates, available both at the server and the clients, can help reduce the communication cost drastically by playing the role of side information.

KLMS is designed to reduce the communication cost in FL by taking advantage of the above observations. It relies on common randomness between the clients and the server in the form of a random SEED (i.e., they can generate the same random samples from the same distribution) and also on the side information available to the server and the clients. Without restricting ourselves to any specific FL framework (we will do this in Section 2.4.3), suppose the server and the clients share a pre-data distribution $p_{\theta^{(t)}}$, and each client has a post-data distribution $q_{\phi^{(t,n)}}$ after the local training steps. As stated in Section 2.4.1, the goal of the server is to compute $\mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N} X^{(t,n)}\right]$ after each round. While this can be done by simply communicating samples $\mathbf{x}^{(t,n)} \sim q_{\phi^{(t,n)}}$, we note that the communicated samples do not need to be the exact same samples that are generated at the client's side. Therefore, instead of communicating a specific realization $\mathbf{x}^{(t,n)} \sim q_{\phi^{(t,n)}}$, KLMS communicates a sample $\mathbf{y}^{(t,n)}$ according to some other distribution $\tilde{q}_{\pi^{(t,n)}}$ such that (i) it is less costly to communicate a sample from $\tilde{q}_{\pi^{(t,n)}}$ rather than $q_{\phi^{(t,n)}}$, and (ii) the discrepancy

$$E = \left|\mathbb{E}_{Y^{(t,n)}\sim\tilde{q}_{\pi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N} Y^{(t,n)}\right] - \mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N} X^{(t,n)}\right]\right| \tag{2.21}$$

is sufficiently small. Motivated by this, each round of KLMS runs as described in Algorithm 2.1. In Theorem 5, we show that the discrepancy in Eq. (2.21) is upper bounded when $K \simeq \exp\{(D_{KL}(q_\phi||p_\theta))\}$. We actually prove it for a general measurable function $f(\cdot)$, for which the discrepancy in Eq. (2.21) is a special case when $f(\cdot)$ is the identity. We note that the previous results on the single-user scenario ($N = 1$) [58, 108] are special cases of our more general framework with N users.

Theorem 5. *Let p_θ and $q_{\phi^{(n)}}$ for $n = 1, \dots, N$ be probability distributions over set $\mathcal{X}$ equipped with some sigma-algebra. Let $X^{(n)}$ be an $\mathcal{X}$-valued random variable with law $q_{\phi^{(n)}}$. Let $r \geq 0$ and $\tilde{q}_{\pi^{(n)}}$ for $n = 1, \dots, N$ be discrete distributions each constructed by $K^{(n)} = \exp\big\{\big(D_{KL}(q_{\phi^{(n)}}||p_\theta) + r\big)\big\}$ samples $\{\mathbf{y}_{[k]}^{(n)}\}_{k=1}^{K^{(n)}}$ from p_θ defining $\pi^{(n)}(k) = \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})}$. Furthermore, for measurable function $f(\cdot)$, let $\|f\|_{\mathbf{q}_\phi} = \sqrt{\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}},\forall n\in[N]}[(\frac{1}{N}\sum_{n=1}^{N} f(X^{(n)}))^2]}$ be its 2-norm under*

$\mathbf{q}_\phi = q_{\phi^{(1)}}, \ldots, q_{\phi^{(N)}}$ *and let*

$$\epsilon = \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}(\log(q_{\phi^{(n)}}/p_\theta) > D_{KL}(q_{\phi^{(n)}} \| p_\theta) + r/2)} \right)^{1/2}. \tag{2.22}$$

Defining $\tilde{q}_{\pi^{(n)}}$ *over* $\{\mathbf{y}_{[k]}^{(n)}\}_{k=1}^{K^{(n)}}$ *as* $\tilde{q}_{\pi^{(n)}}(\mathbf{y}) = \sum_{k=1}^{K^{(n)}} \pi^{(n)}(k) \cdot \mathbf{1}(\mathbf{y}_{[k]}^{(n)} = \mathbf{y})$*, it holds that*

$$\mathbb{P}\left(\left| \mathop{\mathbb{E}}_{Y^{(n)} \sim \tilde{q}_{\pi^{(n)}}, \forall n} \left[\frac{1}{N} \sum_{n=1}^{N} f(Y^{(n)}) \right] - \mathop{\mathbb{E}}_{X^{(n)} \sim q_{\phi^{(n)}}, \forall n} \left[\frac{1}{N} \sum_{n=1}^{N} f(X^{(n)}) \right] \right| \geq \frac{2\|f\|_{\mathbf{q}_\phi} \epsilon}{1-\epsilon} \right) \leq 2\epsilon, \tag{2.23}$$

See Appendix 2.7.2 for the proof. This result implies that when $K^{(n)} \simeq \exp\left(D_{KL}(q_{\phi^{(t,n)}} \| p_{\theta^{(t)}})\right)$, the discrepancy in Eq. (2.21) is small. In practice, we work on blocks of parameters such that $D_{KL}(q_{\phi^{(t,n)}} \| p_{\theta^{(t)}})$ for each block is the same for all clients $n \in [N]$. Hence, we omit the superscript (n) from $K^{(n)}$ and denote the number of samples by K for each client. In Section 2.4.4, we experiment on a toy model and observe that, for a fixed K, the discrepancy in Eq. (2.21) gets smaller as the number of clients N increases, gaining from the participation of more clients in each round.

Adaptive Block Selection for Optimal Bit Allocation

Prior works that have applied importance sampling for Bayesian neural network compression [108], or for differentially private communication in FL [240] split the model into several fixed-size blocks of parameters, and compress each block separately and independently to avoid the high computational cost – which exponentially increases with the number of parameters d. After splitting the model into fixed-size blocks with S parameters each, the authors in [108, 240] choose a single fixed K (number of samples generated from $p_\theta^{(t)}$) for each block no matter what the KL divergence is for different blocks. This yields the same bitrate $\frac{\log K}{S}$ for every model parameter. Furthermore, in [240] the same K throughout training is used without considering the variation in KL divergence over rounds. However, as illustrated in Figure 2.15, KL divergence changes significantly across different layers of the model and across rounds. Hence, spending the same bitrate $\frac{\log K}{S}$ for every parameter at every round is highly suboptimal since it breaks the condition in Theorem 5.

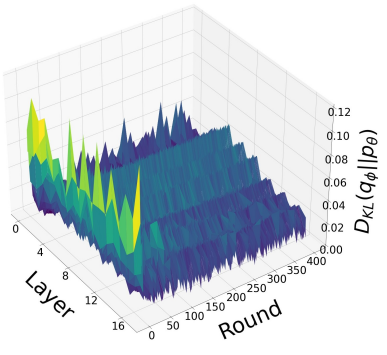


Figure 2.15: Average KL divergence between local post-data distributions of clients and the global pre-data distribution, for different layers and rounds (`FedPM` [200] is used to train `CONV6` on CIFAR-10).

To fix this, we propose an adaptive block selection mechanism, where the block size is adjusted such that the KL divergence for each block is the same and equal to a target value, D_{KL}^{target}. This way, the optimal K for each block is the same and

approximately equal to D_{KL}^{target}, and we do not need to set the block size S ourselves, which was a hyperparameter to tune in [108, 240]. Different from the fixed-size block selection approach in [108, 240], the adaptive approach requires describing the locations of the adaptive-size blocks, which adds overhead to the communication cost. However, exploiting the temporal correlation across rounds can make this overhead negligible. More specifically, we first let each client find their adaptive-size blocks, each having KL divergence equal to D_{KL}^{target}, in the first round. Then the clients communicate the locations of these blocks to the server, which are then aggregated by the server to find the new global indices to be broadcast to the clients, i.e., federated aggregation of block locations. At later rounds, the server checks if, on average, the new KL divergence of the previous blocks is still sufficiently close to the target value D_{KL}^{target}. If so, the same adaptive-size blocks are used in that round. Otherwise, the client constructs new blocks, each having KL divergence equal to D_{KL}^{target}, and updates the server about the new locations. Our experiments indicate that this update occurs only a few times during the whole training. Therefore, it adds only a negligible overhead on the average communication cost across rounds. We provide the pseudocodes for `KLMS` with both fixed- and adaptive-size blocks in Section 2.7.1.

2.4.3 Examples of `KLMS` Adaptated to Well-Known Stochastic FL Frameworks

In this section, we provide four concrete examples illustrating how `KLMS` can be naturally integrated into different FL frameworks with natural choices of pre-data and post-data distributions. Later, we present experimental results showing the empirical improvements `KLMS` brings in all these cases. The corresponding pseudocodes are given in Section 2.7.1.

FedPM-KLMS: As described in Section 2.4.1, in `FedPM` [200], the server holds a global probability mask, which parameterizes a probability distribution over the mask parameters – indicating for each model parameter, with what probability it should remain in the subnetwork. Similarly, each client obtains a local probability mask after local training – parameterizing their locally updated probability assignment for each model parameter to remain in the subnetwork. Choosing the global probability mask $\theta^{(t)}$ as the parameters of the pre-data distribution $p_{\theta^{(t)}}$ and the local probability mask $\phi^{(t,n)}$ as the parameters of the post-data distribution $q_{\phi^{(t,n)}}$ is only natural since the goal in [200] is to send a sample from the local probability distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ with as few bits as possible. This new framework, `FedPM-KLMS`, provides 50 times reduction in bitrate over vanilla `FedPM`.

QSGD-KLMS: As explained in detail in Section 2.4.1, `QSGD` [24] is a stochastic quantization method for FL frameworks that train deterministic model parameters, which outperforms many other baselines in the same setting. Focusing on the most extreme case when the number of quantization levels is $s = 1$, we can express the `QSGD` distribution follows:

$$p_{\texttt{QSGD}}(\hat{\mathbf{v}}_i^{(t,n)}) = \begin{cases} \max\left\{\frac{-\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = -\|\mathbf{v}^{(t,n)}\| \\ \max\left\{\frac{\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \|\mathbf{v}^{(t,n)}\| \\ 1 - \max\left\{\frac{-\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, \frac{\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = 0 \end{cases}, \tag{2.24}$$

which is again a very natural choice of post-data distribution $q_{\phi^{(t,n)}}$ since vanilla `QSGD` requires the clients to take a sample from $p_{\texttt{QSGD}}(\cdot)$ in Eq. (2.24) and communicate the deterministic value of that sample to the server. As for the pre-data distribution, exploiting the temporal correlation in FL, we use the empirical frequencies of the historical updates the server received in the previous round. In other words, in every round t, the server records how many clients communicated a negative value (corresponding to $-\|\mathbf{v}^{(t,n)}\|$), a positive value (corresponding to $\|\mathbf{v}^{(t,n)}\|$), or 0 per coordinate, and constructs the pre-data distribution $p_{\theta^{(t)}}$ from these empirical frequencies for the next rounds. This new framework, `QSGD-KLMS`, yields 12 times reduction in bitrate over vanilla `QSGD`.

SignSGD-KLMS: Since `SignSGD` [44] is not a stochastic quantizer, we first introduce some stochasticity to the vanilla `SignSGD` algorithm and then integrate `KLMS` into it. Instead of mapping the updates to their signs ± 1 deterministically as in vanilla `SignSGD`, the stochastic version we propose does this mapping by taking a sample from the following `SignSGD` distribution

$$p_{\texttt{SignSGD}}(\hat{\mathbf{v}}_i^{(t,n)}) = \begin{cases} \text{Sigmoid}(\frac{\mathbf{v}_i^{(t,n)}}{M}) & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = 1 \\ 1 - \text{Sigmoid}(\frac{\mathbf{v}_i^{(t,n)}}{M}) & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = -1 \end{cases}, \tag{2.25}$$

for some $M > 0$. Instead of taking a sample from $p_{\texttt{SignSGD}}(\cdot)$ and sending the deterministic value of the sample by spending 1 bpp, we can take advantage of the sign symmetry in the model update (about half of the coordinates have positive/negative signs in the update) and reduce the communication cost. For this, we choose $p_{\texttt{SignSGD}}(\cdot)$ in Eq. (2.25) as the post-data distribution $q_{\phi^{(t,n)}}$, and the uniform distribution $U(0.5)$ from the support $\{-1, 1\}$ as the pre-data distribution $p_{\theta^{(t)}}$. This new method, `SignSGD-KLMS`, achieves higher accuracy than vanilla `SignSGD` with 60 times smaller bitrate.

SGLD-KLMS: From the Bayesian FL family, we focus on the recent `SGLD` framework [253] as an example since it provides state-of-the-art results. As explained in detail in Section 2.4.1, due to the stochastic formulation of the Bayesian framework, it is natural to choose the local posterior distributions as the post-data distribution $q_{\phi^{(t,n)}}$, and the global posterior distribution at the server as the pre-data distribution $p_{\theta^{(t)}}$. While extending the existing `SGLD` algorithm (see Section 2.4.1) with `KLMS`, we inject Gaussian noise locally at each client and scale it such that when all the samples are averaged at the server, the aggregate noise sample $\xi^{(t)}$ (see Eq. (2.20)) is distributed according to $\mathcal{N}(0, \boldsymbol{I}_d)$ (more details in Section 2.7.1). This new framework, `SGLD-KLMS`, provides both accuracy and bitrate gains over `QLSD` [253] – the state-of-the-art compression method for

Federated SGLD.

2.4.4 Experiments

Table 2.5: LeNet architecture for MNIST experiments.

Name	Component
conv1	[5 × 5 conv, 20 filters, stride 1], ReLU, 2 × 2 max pool
conv2	[5 × 5 conv, 50 filters, stride 1], ReLU, 2 × 2 max pool
Linear	Linear 800 → 500, ReLU
Output Layer	Linear 500 → 10

We empirically demonstrate the accuracy and bitrate improvements obtained with KLMS by focusing on four KLMS adaptations we covered in Section 2.4.3. We consider four datasets: CIFAR-10 [142], CIFAR-100 [142], MNIST [77], and EMNIST [66] (with 47 classes). For CIFAR-100, we use ResNet-18 [111]; for CIFAR-10, a 6-layer CNN CONV6; for MNIST a 4-layer CNN CONV4 and LeNet; and for EMNIST, again CONV4. Specifically, details on the CNNs can be found in in Table 2.1, while Table 2.4 and Table 2.5 summarize the architectures of ResNet-18 and LeNet, respectively. We first compare FedPM-KLMS, QSGD-KLMS, and SignSGD-KLMS with FedPM [200], QSGD [24], SignSGD [44], TernGrad [262], DRIVE [246], EDEN [247], FedMask [153], and DP-REC [240] on non-Bayesian FL setting. We then provide a comparison of SGLD-KLMS with QLSD [253] on the Bayesian FL scenario. Finally, we present a key ablation study to show how the adaptive block selection strategy in Section 2.4.2 optimizes the bit allocation and helps achieve a smaller bitrate. In the non-Bayesian experiments, clients performed three local epochs with a batch size of 128 and a local learning rate of 0.1; while in the Bayesian experiments, they performed one local epoch. We conducted our experiments on NVIDIA Titan X GPUs on an internal cluster server, using 1 GPU per one run. During non-i.i.d. data split, we choose the size of each client's dataset $|\mathcal{D}^{(n)}| = D_n$ by first uniformly sampling an integer j_n from $\{10, 11, \dots, 100\}$. Then, a coefficient $\frac{j_n}{\sum_n j_j}$ is computed, representing the size of the local dataset D_n as a fraction of the full training dataset size. Moreover, we impose a maximum number of different labels, or classes, $c_{\max}$, that each client can see. This way, highly unbalanced local datasets are generated.

Non-Bayesian Federated Learning

i.i.d. Data Split: For the i.i.d. dataset experiments in Figure 2.16, we set the number of clients to $N = 10$ and consider full client participation. As can be seen from Figure 2.16, FedPM-KLMS and SignSGD-KLMS provide 50 times reduction in communication cost compared to FedPM and SignSGD, respectively (together with the accuracy boost over vanilla SignSGD). QSGD-KLMS, on the other hand, reduces the communication cost by 12 times over vanilla QSGD. Overall, among our baselines, QSGD requires the smallest bitrate, and FedPM achieves the highest accuracy. Surprisingly, FedPM-KLMS requires 10 times smaller bitrate than QSGD while achieving the same accuracy as FedPM at the same time – consistently in all the experiments. The consistent and significant improvements over DP-REC (in both bitrate and accuracy) justify the importance of (i) carefully

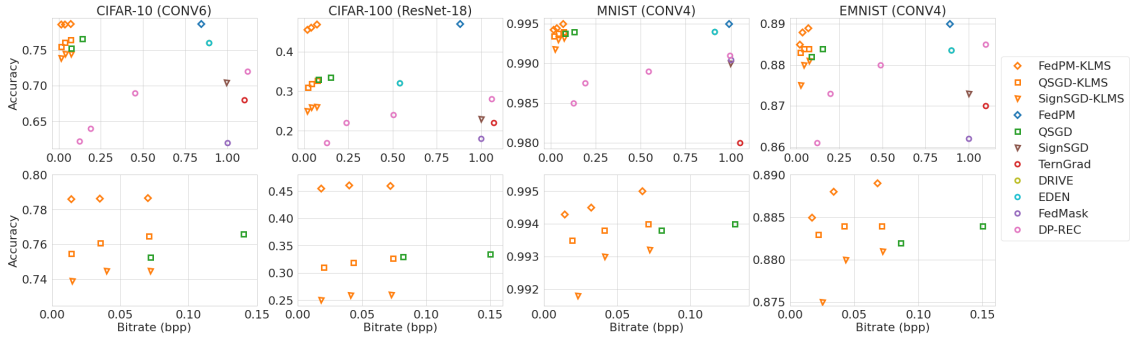


Figure 2.16: `FedPM-KLMS`, `QSGD-KLMS`, and `SignSGD-KLMS` against `FedPM` [200], `QSGD` [24], `SignSGD` [44], `TernGrad` [262], `DRIVE` [246], `EDEN` [247], `FedMask` [153], and `DP-REC` [240] with i.i.d. split and full client participation. The bottom row replicates the upper row zoomed into lower bitrates.

choosing the pre-data and post-data distributions, and (ii) the adaptive block selection that optimizes the bit allocation.

Non-i.i.d. Data Split: For the non-i.i.d. experiments in Figure 2.17, we only compare against the best of our baselines from the i.i.d. results – namely `FedPM`, `QSGD`, `DRIVE`, `EDEN`, and `DP-REC`. We set the number of clients to $N = 100$ and let randomly sampled 20 of them participate in each round. In this scenario, local datasets contain at most $c_{\max}$ different classes, and their sizes are highly skewed. In the experiments in Figure 2.17, we set $c_{\max} = 20$ for CIFAR-100 and $c_{\max} = 4$ for CIFAR-10. Figure 2.17 shows similar gains over the baselines as the i.i.d. experiments in Figure 2.16; in that, `KLMS` adaptations provide up to 50 times reduction in the communication cost compared to the baselines with final accuracy as high as the best baseline. This indicates that the statistical heterogeneity level in the data split, while reducing the performance of the underlying training schemes, does not affect the improvement brought by `KLMS`. We later corroborate this observation with additional experiments.

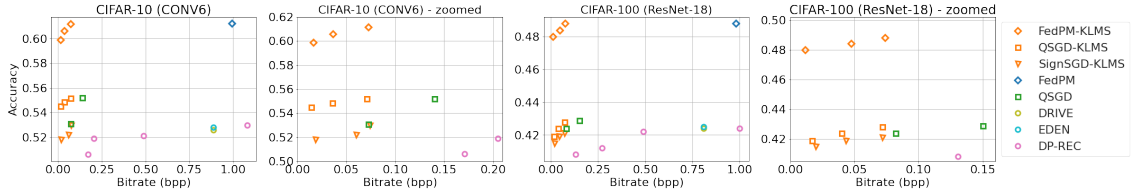


Figure 2.17: `FedPM-KLMS`, `QSGD-KLMS`, and `SignSGD-KLMS` against `FedPM` [200], `QSGD` [24], `DRIVE` [246], `EDEN` [247], and `DP-REC` [166] with non i.i.d. split and 20 out of 100 clients participating every round.

In the end in Figure 2.18 we give the results on `CONV6` and `ResNet-18` on non-i.i.d. CIFAR-10 with $c_{\max} = 2$ and CIFAR-100 with $c_{\max} = 20$, respectively. In both experiments, 10 clients out of 100 clients participate in each round. It is seen that similar accuracy and bitrate improvements are observed to the non-i.i.d. results in Figure 2.17.

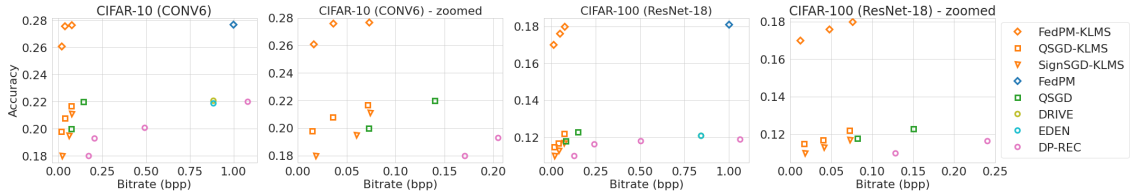


Figure 2.18: Comparison of `FedPM-KLM`, `QSGD-KLM`, and `SignSGD-KLM` with `FedPM` [200], `QSGD` [24], `DRIVE` [246], `EDEN` [247], and `DP-REC` [166] with non i.i.d. split and 10 out of 100 clients participating every round.

Bayesian Federated Learning

We present the comparison of `SGLD-KLMS` with `QLSD` [253] in Figure 2.19-(left). We consider i.i.d. data split and full client participation with the number of clients $N = 10$. It is seen that `SGLD-KLMS` can reduce the communication cost by 5 times more than `QLSD` with higher accuracy on MNIST, where in this case the accuracy is a Monte Carlo average obtained by posterior sampling after convergence.

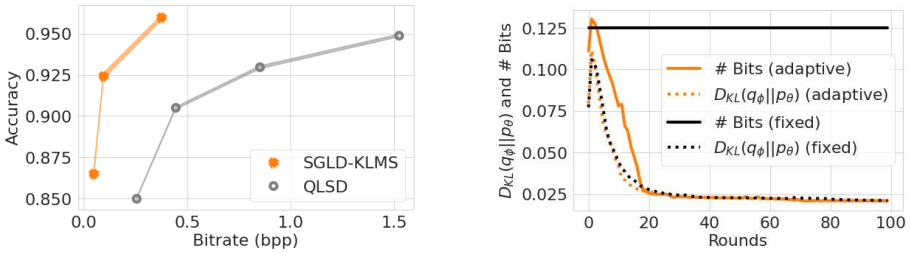


Figure 2.19: **(left)** `SGLD-KLMS` against `QLSD` [253] using LeNet on i.i.d. MNIST dataset. **(right)** `FedPM-KLMS (fixed)` against `FedPM-KLMS (adaptive)` on how well the number of bits approaches the fundamental quantity, KL divergence – using `CONV6` on i.i.d. CIFAR-10. Both KL divergence and the number of bits are normalized by the number of parameters. The final accuracies that `FedPM-KLMS (fixed)` and `FedPM-KLMS (adaptive)` reach differ by only 0.01%.

Ablation Study: The Effect of the Adaptive Bit Allocation Strategy

We conduct an ablation study to answer the following question: *Does adaptive bit allocation strategy really help optimize the bit allocation and reduce # bits down to KL divergence?* To answer this question, in Figure 2.19-(right), we show how the average per-parameter KL divergence and # bits spent per parameter change over the rounds for `FedPM-KLMS` with fixed- and adaptive-size blocks. We adjust the hyperparameters such that the final accuracies differ by only 0.01% on CIFAR-10. For the fixed-size experiments, since we fix K (number of samples per block) and the block size for the whole model and across rounds, # bits per parameter stays the same while the KL divergence shows a decreasing trend. On the other hand, in the adaptive-size experiments, the block size changes across the model parameters and the rounds to guarantee that each block has the same KL divergence. Since all blocks have the same KL divergence, we spend the same

bits for each block as suggested by Theorem 5, which adaptively optimizes the bitrate towards the KL divergence. This is indeed justified in Figure 2.19-(right) since the # bits curve quickly approaches the KL divergence curve.

Additional Experimental Results

`KLMS` on a Toy Model We provide additional insights on `KLMS` employed in a distributed setup similar to that of FL. Specifically, we design a set of experiments in which the server keeps a pre-data distribution $p = \mathcal{N}(0, 1)$, and N clients need to communicate samples according to their local post-data distributions $\{q^{(n)}\}_{n=1}^{N} = \{\mathcal{N}\left(\mu^{(n)}, 1\right)\}_{n=1}^{N}$, which are induced by a global and unknown distribution $q = \mathcal{N}(\mu, 1)$. Each client n applies `KLMS` (see Algorithm 2.1) to communicate a sample $x^{(n)}$ from $q^{(n)}$ using as coding distribution the pre-data distribution p. The server then computes $\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x^{(n)}$ to estimate μ. We study the effect of N, i.e., the number of clients communicating their samples, on the estimation of μ in different scenarios by varying the rate adopted by the clients and the complexity of the problem.

The effect of the overhead r In this example, we simulate an i.i.d. data split by providing all the clients with the same local post-data distribution $q^{(n)} = \mathcal{N}(0.8, 1)\ \forall n \in [N]$. We analyze the bias in the estimation of μ by computing a Monte Carlo average of the discrepancy in Eq. (2.21) (see Figure 2.20-(**right**)), together with its empirical standard deviation (see Figure 2.20-(**left**)). From Figure 2.20, we can observe that, as conjectured, the standard deviation of the gap decreases when N increases, meaning that the estimation is more accurate around its mean value, which is also better for larger values of N. Also, as expected, a larger value of the overhead r induces better accuracy.

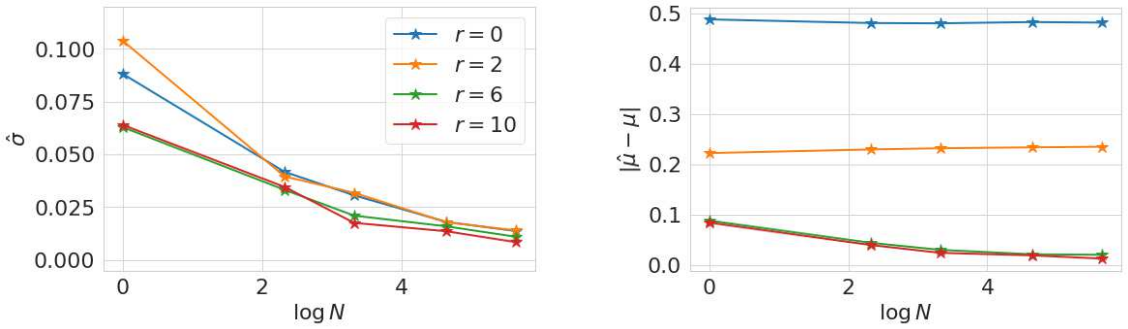


Figure 2.20: Estimation gap statistics for different values of r, as a function of the number of participating clients N. (**left**) The empirical standard deviation of the estimation gap, computed over 100 runs. (**right**) Estimation gap between μ and $\hat{\mu}$ averaged over 100 runs.

The effect of non-i.i.d. data split In this other set of experiments we simulate a non-i.i.d. data split by inducing, starting from the same pre-data distribution p, different local post-data distributions, simulating drifts in updates statistics due to data heterogeneity. Specifically, we

set again $\mu = 0.8$, and then, $\forall n \in [N]$, $\mu^{(n)} = 0.8 + u^{(n)}$, where $u^{(n)} \sim \text{Unif}([-\eta, \eta])$, for $\eta \in \{0.05, 0.1, 0.25, 0.4\}$. In all experiments, $r = 6$. As we can see from Figure 2.21, when N is very small (~ 1), then high level of heterogeneity in the update statistics can indeed lead to poor estimation accuracy. However, for reasonable values of N, this effect is considerably mitigated, suggesting that for real-world applications of FL, where the number of devices participating to each round can be very large, `KLMS` can still improve state-of-the-art compression schemes by large margin.

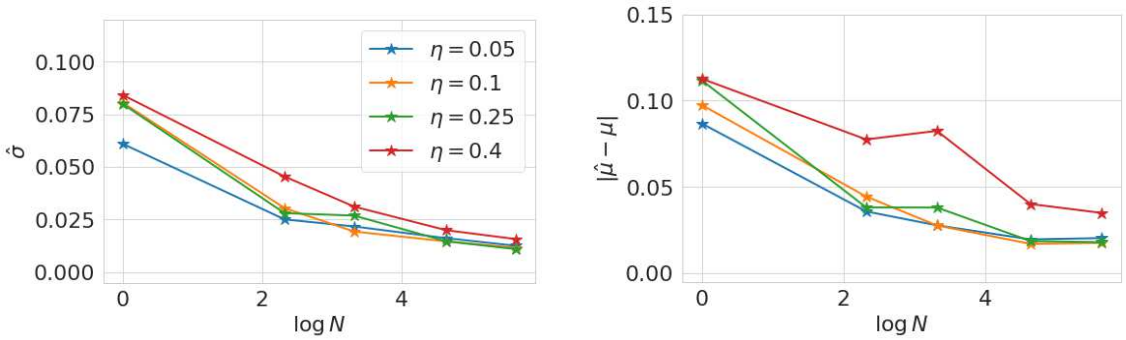


Figure 2.21: Estimation gap statistics for different values of η, as a function of the number of participating clients N. (**left**) The empirical standard deviation of the estimation gap, computed over 100 runs. (**right**) Estimation gap between μ and $\hat{\mu}$ averaged over 100 runs.

2.4.5 Discussion & Conclusion

We introduced KL Divergence Minimization with Side Information (`KLMS`) – a recipe for reducing the communication cost in stochastic FL frameworks by exploiting the side information available to the server and correlated with the local model updates. We highlighted the existence of highly natural choices of pre-data distribution (side information at the server) and post-data distribution (at the clients) in FL that we can take advantage of to reduce the communication cost significantly. Moreover, we showed how to adaptively adjust the bitrate across the model parameters and training rounds to achieve the fundamental communication cost – the KL divergence between the pre-data and post-data distributions. While we showed four `KLMS` adaptations that reduce the communication cost 50 times more than our baselines and still preserve (sometimes improve) the accuracy, it can be adapted to many other stochastic FL frameworks and can provide similar communication gains.

2.5 Semantic Communications for Learnable Concepts

2.5.1 Introduction and Motivation

With the growing number of mobile devices and sensors, massive amounts of data are collected today at the edge of communication networks. On the one hand, this data is the fuel for training

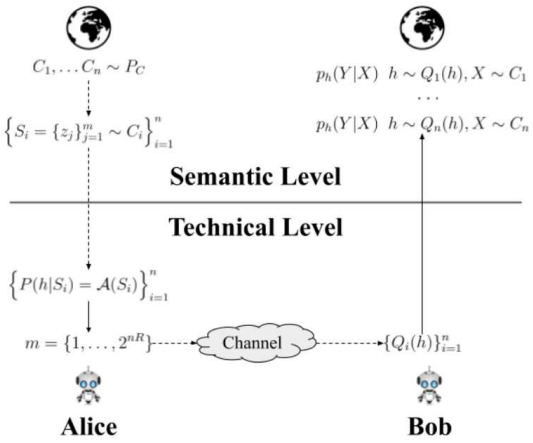


Figure 2.22: The problem of communicating concepts.

large learning models like DNNs; on the other hand, these models need to be stored, compressed, and communicated over bandwidth limited channels to the cloud, and protected against security and privacy risks [267]. These issues are increasingly limiting the application of typical centralized training approaches. As we have seen throughout the chapter, various federated/distributed learning paradigms have emerged as potential solutions to mitigate these limitations, which allow the models to be locally trained, and then aggregated in a cloud or edge server without moving local private data [171]. The main paradigm shift in distributed learning is to move the models, rather than the data, throughout the network, providing better privacy guarantees and reducing the communication load. However, today even the sizes of such learned models are becoming a concern, as transmitting huge models back and forth for training or inference purposes can easily congest wireless networks, specifically when considering that the edge devices like mobile phones, cars, robots etc., are usually wirelessly connected to the network, and thus have limited bandwidth [124].

Consequently, it is time to investigate, with the proper information-theoretic models and tools, the fundamental limits of communicating models over rate-limited channels, and not just raw data. To this end semantic communications, which concerns with the semantic aspect of the message, maps naturally to the transmission of these learning models [100]. The communication fidelity of these models can be judged by how close the behavior of the reconstructed model at the receiver is to the desired one, rather than by the accuracy of the reconstruction in the parameter space [124].

2.5.2 System Model

Let $\mathcal{E}$ denote the environment, i.e., the source, that generates a sequence of n concepts, e.g., tasks, $\{c_i\}_{i=1}^n$, $c_i \in \mathcal{C}$, sampled with probability P_C in an i.i.d. fashion. While P_C is known by both Alice and Bob, neither of them can observe the sampled concepts directly. Alice has access to a sequence of m samples $\{z_{i,j}\}_{j=1}^m$, where $z_{i,j} = (x_{i,j}, y_{i,j}) \in \mathcal{Z}$, sampled according to each of the

concept distributions $p_{c_i}(Y|X)p_{c_i}(X),\ \forall i = 1, \ldots, n$. Alice and Bob agree on a hypothesis class, i.e., the model class $\mathcal{H}$, and on a pre-data coding probability distribution $P_h,\ \forall h \in \mathcal{H}$. We call the sequence $\{z_{i,j}\}_{j=1}^m$ of samples the dataset s_i. Alice applies a learning algorithm $\mathcal{A} : \mathcal{Z}^m \to \Phi(\mathcal{H})$ on s_i, which is a possibly stochastic function mapping a dataset to a probability distribution $Q_{h|s_i} = \mathcal{A}(s_i)$ over the set of models $\mathcal{H}$, and so, over the subset of all possible probability mappings $h : \mathcal{X} \to \Phi(\mathcal{Y})$, representing Alice's concept belief. With $\Phi(\mathcal{X})$ we denote the set of all possible probability distributions over the set $\mathcal{X}$. Consequently, the models are functions used by Alice and Bob to represent (or, more precisely, to approximate) the concepts' relation among data. To measure how well a model h approximates a concept c, a per-sample loss $\ell_c(h, z) : \mathcal{H} \times \mathcal{Z} \to [0, 1]$ is defined, which compares the discrepancy between $p_c(y|x)$ and $h(x)$. In this work, we assume a bounded loss within $[0, 1]$ for the sake of clarity of exposition. Then, $\ell_c(Q, z) : \Phi(\mathcal{H}) \times \mathcal{Z} \to [0, 1]$ is the performance of the model belief Q, which is defined as $\ell_c(Q, z) = \mathbb{E}_{h\sim Q}[\ell_c(h, z)]$. Upon observing the data, Alice can compute her empirical performance by using the *empirical loss* on her dataset S as

$$\tilde{\mathcal{L}}_C(Q, S) = \frac{1}{|S|} \sum_{z\in S} \ell_C(Q, z), \tag{2.26}$$

where $Q = Q_{H|S} = \mathcal{A}(S)$ is the post-data distribution inferred by Alice, given the data. We assume that, for any sequences of datasets s^n, s'^n, Alice's distribution can be factorized as $Q^n_{h^n|s^n} = \prod_{i=1}^n Q_{h_i|s_i}$ such that $s_i = s'_j \Rightarrow Q_{h_i|s_i} = Q_{h_j|s'_j}$. However, to assess how well the belief Q represents the concept C, in machine learning we are usually interested in the *true loss*

$$\mathcal{L}_C(Q) = \mathbb{E}_{Z\sim C}[\ell_C(Q, Z)], \tag{2.27}$$

i.e., the expected performance on a new unseen sample. Given the realization of the datasets $\{s_i\}_{i=1}^n$, the problem for Alice is then to convey a message to Bob through a constrained communication channel, which limits the maximum number of bits she can convey per model, so that Bob can use the received information to reconstruct models $\{\hat{h}_i\}_{i=1}^n$ that can approximate the concepts $\{c_i\}_{i=1}^n$ by minimizing the loss on random samples $\{z_i\}_{i=1}^n$ distributed according to the sequence of concepts, i.e., the true loss in Equation Eq. (2.27). We can observe that the task for Bob is not to exactly reconstruct the sequence $\{h_i\}_{i=1}^n$ sampled by Alice, but rather to obtain samples $\{\hat{h}_i\}_{i=1}^n$ whose probability distributions are close to the target ones, i.e., $\{Q_{h_i|s_i}\}_{i=1}^n$.

Remark. We now briefly discuss why we are interested in learning rules $\mathcal{A}(S)$ that output model distributions, rather than single-point solutions:

- First of all, the case in which Alice finds a point-wise estimate of the best model h^* is included as a special case $Q_{h|S} = \delta_{h^*}$.

- Alice may want to express her uncertainty around the best choice h^*, which may be intrinsic in the learning algorithm $\mathcal{A}$, through the distribution $Q_{h|S}$.

- Usually, optimization algorithms used to train DNNs, like SGD, are stochastic algorithms.

- When $\mathcal{H}$ is the set of all DNNs h_ω with a specific architecture parameterized by the parameter vector ω, there exist many vectors ω performing in the same way. Moreover, small perturbations to the parameters usually does not reduce the final performance. This means that it is not required for Bob to reconstruct the exact value of the network parameters, but rather a nearby or an equivalent solution, and this variability is represented by $Q_{h_\omega|S}$. More importantly, $Q_{h_\omega|S}$ can be exploited to reduce the rate needed to convey the models, thus saving network resources [108]. This is the semantic aspect of communication captured by our framework, as the meaning of a concept c, i.e., the real unknown mapping, is conveyed through the model belief $Q_{h|S}$, whose loss expressed in Equation Eq. (2.27) quantifies its fidelity with respect to the real concept c.

2.5.3 The Rate-Distortion Characterization

In this section, we first characterize the limit of the problem when $n = 1$, i.e., one-shot concept communication, and then generalize the problem to the n-sequence formulation. For the latter, two kinds of performance metrics are defined: the first one provides average performance guarantees, while the second one ensures the same performance guarantee for each sample $\hat{h}$. We will show that the minimum achievable communication rate that can guarantee a certain distortion level is the same in both cases, as long as sufficient common randomness between Alice and Bob is available.

Single-Shot Problem

The single-shot version of the problem has been studied in [108], where the authors propose `MIRACLE`, a neural network compression framework based on bits back coding [95], providing an efficient single-shot model compression scheme showing empirically that, with enough common randomness, it is possible to convey the model with an average of $K \simeq D_{\mathrm{KL}}(Q||P)$ bits with very good performance, where P is the pre-data coding model distribution, and Q is the optimized post-data distribution, providing a belief over well-performing neural networks, or, equivalently, over a set of parameter vectors, as explained in Section 2.5.2. However, from Lemma 1.5 in [105], the average number of bits $\mathbb{E}_{C,S}[K]$ needed to exactly code Q with P, when sufficient common randomness is available, can be bounded by

$$\mathcal{R} \leq \mathbb{E}_{C,S}[K] \leq \mathcal{R} + 2\log(\mathcal{R} + 1) + \mathcal{O}(1), \tag{2.28}$$

where $\mathcal{R} = \mathbb{E}_{C,S}[D_{\mathrm{KL}}(Q||P)]$, while using exactly $\mathcal{R}$ bits may lead to samples which are distributed according to $\tilde{Q}$, slightly different from the target Q [105]. More recent results [155] allow to find even stronger guarantees (Corollary 3.4 in [235]) for this relationship:

$$\mathbb{E}_{C,S}[K] \leq \mathcal{R} + \log(\mathcal{R} + 1) + 4. \tag{2.29}$$

n-Length Formulation

We now study the problem depicted in Figure 2.22, when we let Alice code a sequence of n concept realizations, i.e., datasets, and study the information-theoretic limit of the system as $n \to \infty$. Specifically, we are interested in the trade-off between the rate R, which is defined as the average number of bits consumed per model by Alice to convey the concept process to Bob, and the performance, which is the true loss that can be obtained by Bob (see Eq. (2.27)). We start by defining the proper quantities involved.

Definition 6 (Rate-Distortion Coding Scheme)**.** *A $(2^{nR}, n)$ coding scheme consists of an alphabet $\mathcal{X}$, a reconstruction alphabet $\hat{\mathcal{X}}$, an encoding function $f_n : \mathcal{X}^n \to \{1, 2, \dots, 2^{nR}\}$, a decoding function $g_n : \{1, 2, \dots, 2^{nR}\} \to \hat{\mathcal{X}}^n$, and a distortion measure $d : \mathcal{X}^n \times \hat{\mathcal{X}}^n \to \mathbb{R}^+$, comparing the fidelity between x^n and $\hat{x}^n$. Specifically, we are interested in the expected distortion $\mathbb{E}\left[d(x^n, \hat{x}^n)\right] = \sum_{x^n \in \mathcal{X}^n} p(x^n) d(x^n, g_n(f_n(x^n)))$.*

Definition 7 (Rate-Distortion)**.** *A rate-distortion pair (R, ϵ) is said to be achievable for a source $p(x)$ and a distortion measure d, if there exists a sequence of $(2^{nR}, n)$ rate-distortion coding schemes with*

$$\lim_{n \to \infty} \mathbb{E}\left[d(x^n, g_n(f_n(x^n)))\right] < \epsilon. \tag{2.30}$$

However, as specified in the remark in Section 2.5.2, we are interested in conveying beliefs $Q \in \Phi(\mathcal{H})$, i.e., samples drawn according to the probability Q, obtained from the datasets $s^n \in \mathcal{S}^n$, where $\mathcal{S}^n = \mathcal{Z}^m$. In our case, the coding function $f_n : \mathcal{S}^n \to \{1, 2, \dots, 2^{nR}\}$ maps the sequence $s^n = \{s_i\}_{i=1}^n$ to a message $f_n(s^n)$ from which Bob can obtain the models $\hat{h}^n = g_n(f_n(s^n))$. Our distortion then considers the difference between the $\hat{h}^n$'s distribution $\hat{Q}^n$, and the one achievable by Alice $Q^n = \{\mathcal{A}(s_i)\}_{i=1}^n$ by comparing their samples $\hat{h}^n$ and h^n. Consequently, we define with $\hat{Q}_{S^n, \hat{H}^n}$ (or simply $\hat{Q}^n$) the joint distribution between the datasets and models induced by a $(2^{nR}, n)$ coding scheme, whose marginals are $\hat{Q}_{S_i, \hat{H}_i}$ for $i = 1, \dots, n$.

Definition 8 (Concept Distortion)**.** *For the problem of communicating concepts, we define the following distortion on the model beliefs Q and $\hat{Q}$:*

$$d_{sem}(Q, \hat{Q}) = \mathbb{E}_{C,S}\left[\mathcal{L}_C(\hat{Q}) - \mathcal{L}_C(Q)\right]. \tag{2.31}$$

The rationale behind this definition is that Q, which is the target distribution at the transmitter, is optimized on a given dataset S without any constraint. Therefore, it is reasonable to assume $\mathcal{L}_C(\hat{Q}) - \mathcal{L}_C(Q)$ to be always non-negative. We notice that d_{sem} quantifies the gap between the concept reconstruction at the receiver and the one at the transmitter, which is a semantic measure on the unknown true loss $\mathcal{L}(Q)$.

Definition 9 (n-Sequence Concept Distortion)**.** *For the problem of communicating concepts, we*

define the following distortion on the sequence of the model beliefs Q^n and $\hat{Q}^n$:

$$d_{avg}(Q^n, \hat{Q}^n) = \frac{1}{n}\sum_{i=1}^{n} d_{sem}(Q_i, \hat{Q}_i), \tag{2.32}$$

and

$$d_{max}(Q^n, \hat{Q}^n) = \max_{i=\{1,\dots,n\}} d_{sem}(Q_i, \hat{Q}_i). \tag{2.33}$$

In practice, d_{avg} defines a constraint on the true loss achievable by Bob averaged over the performance of the sequence $\hat{h}^n \sim \hat{Q}^n$, whereas d_{max} imposes a constraint on the loss of every marginal $\hat{h}_i \sim \hat{Q}_i^n,\ i \in \{1, \dots, n\}$.

Remark. The distortion, easily defined for one-shot communications, can be generalized to a sequence of n concepts in multiple ways. Specifically, if one is interested in a system-level loss, then satisfying the constraint on d_{avg} could be enough. However, to provide a per-model guarantee on the performance, then d_{max} is the distortion to use.

Definition 10 (Rate-Distortion Region)**.** *The rate-distortion region for a source is the closure of the set of achievable rate-distortion pairs (R, ϵ).*

Definition 11 (Rate-Distortion Function)**.** *The rate-distortion function $R(\epsilon)$ is the infimum of rates R such that (R, ϵ) is in the rate-distortion region.*

Average Distortion d_{avg}

We first analyze the problem with the average distortion d_{avg}, as defined in Equation Eq. (2.32).

Theorem 12 (Rate-Distortion Theorem for d_{avg})**.** *For the problem of communicating concepts with distortion d_{avg}, the rate-distortion function satisfies*

$$R(\epsilon) = \min_{\substack{\tilde{Q}_{H|S}: \\ d_{sem}(Q,\tilde{Q})\leq\epsilon}} I(S; H), \tag{2.34}$$

where $I(S; H)$ is the mutual information between the data S and the model H [69].

Proof. See Appendix 2.7.2 □

Maximum Distortion d_{max}

In this case the distortion function implies a constraint on the performance of each symbol, i.e., model realization. First of all, we just provide a simple scheme in which $\lim_{n\to\infty} d_{avg}(Q^n, \hat{Q}^n) = 0$ does not imply $\lim_{n\to\infty} d_{max}(Q^n, \hat{Q}^n) = 0$, meaning that in general a code that achieves 0 distortion on average, may not achieve 0 distortion model-wise, i.e., we cannot guarantee a single-model performance.

Example 2.6. *Let $\mathcal{H} = \{h_0, h_1\}$, and performance $\ell_c(h_0, z) = 0, \ell_c(h_1, z) = 1,\ \forall z \in \mathcal{Z}, \forall\ c \in \mathcal{C}$. Let $\forall\ c^n \in \mathcal{C}^n$, Alice's distribution $Q_i(h_j|S) = \frac{1}{2}$, where $j \in \{0, 1\}$, and $Q^n = \prod_{i=1}^n Q_i$, while Bob's distribution is deterministic $\hat{Q}_{2i}(h_0|S) = 1, \hat{Q}_{2i+1}(h_1|S) = 1$. Then*

$$d_{avg}(Q^n, \hat{Q}^n) = \begin{cases} \frac{1}{n}\sum_{i=1}^{\frac{n}{2}}\left(1 - \frac{1}{2} + 0 - \frac{1}{2}\right) = 0, & \text{if } n \text{ is even} \\ \frac{1}{n}[\sum_{i=1}^{\frac{n-1}{2}}\left(1 - \frac{1}{2} + 0 - \frac{1}{2}\right) + 1 - \frac{1}{2}] = \frac{1}{2n}, & \text{otherwise} \end{cases}$$

Thus, $\lim_{n\to\infty} d_{avg}(Q^n, \hat{Q}^n) = 0$
but

$$d_{max}(Q^n, \hat{Q}^n) = \max\left\{1 - \frac{1}{2}, 0 - \frac{1}{2}\right\} = \frac{1}{2}$$
$$\implies \lim_{n\to\infty} d_{max}(Q^n, \hat{Q}^n) \neq 0.$$

The question now is what is needed to ensure the same distortion ϵ to the single-model performance, i.e., to ensure $d_{max} < \epsilon$, which is of particular interest in the semantic communication of concepts like the one considered here. We now distinguish between the two ways in which $\hat{Q}^n$ can converge to a target Q^n – empirical and strong. These two notions, introduced later, map precisely onto the difference between convergence of d_{avg} and d_{max}. The focus is changed from determining which distortion is achievable to what joint distributions of H and S are feasible under some rate constraint.

First, we extend the encoding and decoding functions f_n and g_n in Definition 6 to accept an additional common input $\omega \in \Omega$, which is generated by a source of common randomness $p(\omega)$. We define a $(2^{nR}, 2^{nR_0}, n)$ stochastic code consisting of functions $f_n : \mathcal{S}^n \times \{1, \ldots, 2^{nR_0}\} \to \{1, 2, \ldots, 2^{nR}\}$ and $g_n : \{1, 2, \ldots, 2^{nR}\} \times \{1, \ldots, 2^{nR_0}\} \to \hat{\mathcal{H}}^n$, which consumes on average R_0 bits of common randomness per sample.

Definition 13. *A desired distribution $Q_{S,H}$ is achievable for empirical coordination with rate pair (R, R_0) if there exists a sequence of $(2^{nR}, 2^{nR_0}, n)$ codes and a choice of common randomness distribution $p(\omega)$ such that*

$$TV\left(\hat{Q}_{s^n\hat{h}^n}, Q_{S,H}\right) \to 0, \tag{2.35}$$

where $\hat{Q}_{s^n\hat{h}^n}(s, \hat{h}) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}_{(s_i,\hat{h}_i)=(s,\hat{h})}$, $TV(\hat{Q}, Q)$ indicates the total variation between distributions $\hat{Q}$ and Q, and $\mathbf{1}_{\mathcal{I}} = 1$ if the condition $\mathcal{I}$ is true, and 0 otherwise.

In other words, the empirical coordination property requires that the joint empirical distribution of the pairs $(s_i, g_n(f_n(s^n))_i)$ converges, in total variation, to the desired distribution. Notice how, in Example 2.6, the joint distributions Q^n and $\hat{Q}^n$ converge in their empirical distributions, while differing letter-wise.

As already pointed out in [71], introducing common randomness does not improve the performance of empirical coordination schemes, meaning that any distribution achievable for empirical

coordination by a $(2^{nR}, 2^{nR_0}, n)$ coding scheme is also achievable with $R_0 = 0$. Moreover, empirical coordination schemes can be used to construct rate-distortion schemes for d_{avg} [71]. However, as observed in Example 2.6, they do not equate to the same per-symbol performance requirements.

Definition 14. *A desired distribution $Q_{D,H}$ is achievable for strong coordination with rate pair (R, R_0) if there exists a sequence of $(2^{nR}, 2^{nR_0}, n)$ coordination codes and a choice of common randomness distribution $p(\omega)$ such that*

$$TV\left(\hat{Q}_{s^n\hat{h}^n}, \prod_{i=1}^{n} Q_{s_i,h_i}\right) \to 0, \tag{2.36}$$

where $\hat{Q}_{s^n\hat{h}^n}$ is the joint distribution induced by the stochastic coding scheme.

Lemma 15 (d_{max} Achievability)**.** *With sufficient common randomness, the rate-distortion region (R, ϵ) for d_{max} is the same as the one for d_{avg}.*

Proof. See Appendix 2.7.2. □

Given a constraint on the distortion ϵ for d_{avg} achievable with minimum rate of R_ϵ, if Alice and Bob can use common randomness, then it is possible to satisfy, at the same rate R_ϵ, the same level of distortion ϵ for d_{max}. To translate it into machine learning parlance, we showed that the communication rate needed to provide some performance guarantees on the expected average system test error, and on the expected single-model performance, is the same, as long as sufficient common randomness is available. In both cases, the characterization is over the expected performance, thus for any one realization the k-th model might have higher than desired loss.

Coding Without the Marginal Q_H

We notice that all the previous achievability results assume knowledge of the exact marginal $Q_H = \sum_{c\in\mathcal{C}, s\in\mathcal{Z}^m} Q_{H|s}P_{s|c}P_c$ to be used as pre-data coding model distribution (see Section 2.5.2), which is usually not known and difficult to obtain. Consequently, we are interested in studying the minimum achievable rate, when a generic coding distribution P_H is used to code $Q_{H|S}$.

Theorem 16 (Achievability with General P_H)**.** *For the problem of communicating concepts, the minimum achievable rate for both d_{avg} and d_{max} with pre-data coding distribution P_H is*

$$R(\epsilon) = \min_{\substack{\tilde{Q}_{H|S}: \\ d_{sem}(Q,\tilde{Q})\leq\epsilon}} \mathbb{E}_{C,S}\left[D_{KL}\left(\tilde{Q}_{H|S}\|P_H\right)\right], \tag{2.37}$$

assuming sufficient common randomness.

Proof. See Appendix 2.7.2. □

It is known that when $P_H = Q_H$, i.e., using the marginal, Equation Eq. (2.37) is minimized, and so when the marginal is not known we pay an additional penalty given by $\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}\left(Q_{H|S}\|P_H\right)\right] - I(S;H) = \mathbb{E}_{C,S}\left[D_{\mathrm{KL}}\left(Q_H\|P_H\right)\right]$.

2.6.1 Communicating the Data vs Communicating the Model

To motivate our research problem, we comment on the advantages for Alice of first compressing a trained model, and then sending it to Bob (*scheme 1*), versus a second framework (*scheme 2*), in which Alice communicates a compressed version of the dataset $\hat{S}^2 = \rho(S)$, using which Bob trains his models. Given a quantity X, we indicate with X^i the same quantity in the i-th scheme. First of all, we see that in scheme 1, the corresponding Markov chain is $C \to S \xrightarrow{R} \hat{H}^1$, where the R above the arrow indicates the information bottleneck between the two random variables. However, the same chain for the second scheme reads $C \to S \xrightarrow{R} \hat{S}^2 \to \hat{H}^2$. By the data processing inequality [69], the rate constraint imposes $I(S;\hat{H}) \leq R$ in both cases, limiting the set of all feasible beliefs $Q \in \Phi(\hat{\mathcal{H}})$.

Now, we reasonably assume that the optimal solution $Q^*(S,H)$ constrained to $I(S;H) \leq R$ lies on the boundary of the constraint, i.e., $I_{Q^*}(S;H) = R$, where $I_{Q^*}(S;H)$ indicates that the mutual information is computed using Q^*. In this case we can see that for scheme 1, $I(S;\hat{H}^1) = R$, whereas for scheme 2, $I(S;(\hat{S}^2,\hat{H}^2)) = R$. Indeed, in the former scheme Alice conveys just the random variable $\hat{H}^1$, i.e., the model, to Bob, whereas in the latter, the pair $(\hat{S}^2,\hat{H}^2)$ is communicated. However, the information bottleneck is the same. Consequently, we obtain

$$I(S;\hat{H}^1) = I(S;\hat{H}^2) + I(S;\hat{S}^2|\hat{H}^2). \tag{2.38}$$

By non-negativity of the mutual information, we always have $I(S;\hat{H}^1) \geq I(S;\hat{H}^2)$, meaning that the rate constraint on the communication channel translates differently into a model constraint for the two schemes, being stricter for the second one. In particular, the gap between the two schemes is exactly $I(S;\hat{S}^2|\hat{H}^2)$. Consequently, for scheme 2 the optimal compression function $\rho(S)$ must achieve $I(S;\hat{S}^2|\hat{H}^2) = 0$, and so the only way to match the performance of scheme 1 is to account for the optimal distribution $Q_{\hat{H}|S}$ when computing $\hat{S}^2$.

Distortion Rate Bound

In this section, we bound the distortion-rate function for d_{max}, which is useful to translate the rate constraint into a performance gap. We now define $\Delta_R = R^* - R$ to be the difference between the rate R^* of the optimal distribution and the rate R of the channel imposed by the problem. In general, we assume $R^* \geq R$ and $\mathbb{E}_{C,S}\left[\mathcal{L}(Q^*)\right] \leq \mathbb{E}_{C,S}\left[\mathcal{L}(Q)\right]$, where Q^* is achievable with rate R^*, and Q with rate R.

Lemma 17. *Assuming that $\ell_C(z,h)$ is upper bounded by L_{max} $\forall h \in \mathcal{H}$, $\forall z \in \mathcal{Z}$, $\forall C \in \mathcal{C}$, and that distortion d_{max} is considered, the distortion-rate function for the problem of communicating concepts when using scheme 1 can be upper bounded by*

$$\epsilon^1(\Delta_R) \leq L_{max} \cdot \min\left\{\sqrt{\frac{1}{2}\Delta_R}, \sqrt{1 - e^{-\Delta_R}}\right\}, \tag{2.39}$$

and when using scheme 2 by

$$\epsilon^2(\Delta_R) \leq L_{max} \cdot \min\Big\{\sqrt{\frac{1}{2}\Big(\Delta_R + I(S;\hat{S}^2|\hat{H}^2)\Big)}, \sqrt{1 - e^{-\left(\Delta_R + I(S;\hat{S}^2|\hat{H}^2)\right)}}\Big\}. \tag{2.40}$$

Proof. See Appendix 2.7.2. □

2.6.2 Conclusion

We introduce the problem of conveying concepts, where concepts naturally appear as sequences of samples and can be approximated by learnable models, as in standard statistical learning. We study the framework by applying information-theoretic tools to the problem of communicating many models jointly. We characterize its rate-distortion function for two different notions of system-level distortion, provide a bound for the distortion-rate function, and argue why jointly learning, compressing, and communicating models should be preferred over compressing and sending the datasets.

2.7 Supplementary & Proofs

2.7.1 Pseudocode

`FedPM` Algorithm

We provide the pseudocode for `FedPM` in Algorithms 2.2 and 2.3. In Algorithm 2.3, the prior resetting scheduling policy is controlled by the procedure *ResPrior(t)*, which may depend on quantities other than the round number t, such as loss.

Algorithm 2.2 `FedPM`.

Hyperparameters: learning rate η, minibatch size B, number of local iterations τ.
Inputs: local datasets $\mathcal{D}_i$, $i = 1, \ldots, N$
Output: random seed SEED and binary mask parameters $\boldsymbol{m}^{k,T}$

At the server, initialize a random network with weight vector $\boldsymbol{w}^{\text{init}} \in \mathbb{R}^d$ using a random seed SEED, and broadcast it to the clients.
At the server, initialize the random score vector $\boldsymbol{s}^{g,0} \in \mathbb{R}^d$, and compute $\boldsymbol{\theta}^{g,0} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{g,0})$.
At the server, initialize Beta priors $\boldsymbol{\alpha}^{g,0} = \boldsymbol{\beta}^{g,0} = \boldsymbol{\lambda}_0$.
for $t = 1, \ldots, T$
  Sample a subset $\mathcal{K}_t \subset \{1, \ldots, N\}$ of $|\mathcal{K}_t| = K$ clients without replacement.
  On Client Nodes:
  for $k \in \mathcal{K}_t$
    Receive $\boldsymbol{\theta}^{g,t-1}$ from the server and set $\boldsymbol{s}^{k,t} = \text{Sigmoid}^{-1}(\boldsymbol{\theta}^{g,t-1})$.
    for $l = 1, \ldots, \tau$
      $\boldsymbol{\theta}^{k,t} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{k,t})$
      Sample binary mask $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$.
      $\dot{\boldsymbol{w}}^{k,t} \leftarrow \boldsymbol{m}^{k,t} \odot \boldsymbol{w}^{\text{init}}$
      $\text{grad}_{\boldsymbol{s}^{k,t}} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \nabla \ell(\dot{\boldsymbol{w}}^{k,t}; \mathcal{B}_j^k)$; $\{\mathcal{B}_j^k\}_{j=1}^{B}$ is uniformly chosen from $\mathcal{D}_k$
      $\boldsymbol{s}^{k,t} \leftarrow \boldsymbol{s}^{k,t} - \eta \cdot \text{grad}_{\boldsymbol{s}^{k,t}}$
    end for

    $\boldsymbol{\theta}^{k,t} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{k,t})$
    Sample a binary mask $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$.
    Send the arithmetic coded binary mask $\boldsymbol{m}^{k,t}$ to the server.
  end for

  On the Server Node:
  Receive $\boldsymbol{m}^{k,t}$'s from K client nodes.
  $\boldsymbol{\theta}^{g,t} = \text{BayesAgg}(\ \{\boldsymbol{m}^{k,t}\}_{k \in \mathcal{K}_t},\ t)$ // See Algorithm 2.3.
  Broadcast $\boldsymbol{\theta}^{g,t}$ to all client nodes.
end for

Sample the final binary mask $\boldsymbol{m}^{\text{final}} \sim \text{Bern}(\boldsymbol{\theta}^{g,T})$.
Generate the final model: $\dot{\boldsymbol{w}}^{\text{final}} \leftarrow \boldsymbol{m}^{\text{final}} \odot \boldsymbol{w}^{\text{init}}$.

Algorithm 2.3 BayesAgg.

Inputs: clients' updates $\{\boldsymbol{m}^{k,t}\}_{k\in\mathcal{K}_t}$, and round number t
Output: global probability mask $\boldsymbol{\theta}^{g,t}$

if ResPriors(t)
$\quad\boldsymbol{\alpha}^{g,t-1} = \boldsymbol{\beta}^{g,t-1} = \boldsymbol{\lambda}_0$
end if

Compute $\boldsymbol{m}^{\text{agg},t} = \sum_{k\in\mathcal{K}_t} \boldsymbol{m}^{k,t}$.
$\boldsymbol{\alpha}^{g,t} = \boldsymbol{\alpha}^{g,t-1} + \boldsymbol{m}^{\text{agg},t}$
$\boldsymbol{\beta}^{g,t} = \boldsymbol{\beta}^{g,t-1} + K \cdot \mathbf{1} - \boldsymbol{m}^{\text{agg},t}$
$\boldsymbol{\theta}^{g,t} = \frac{\boldsymbol{\alpha}^{g,t}-1}{\boldsymbol{\alpha}^{g,t}+\boldsymbol{\beta}^{g,t}-2}$
Return $\boldsymbol{\theta}^{g,t}$

`KLMS` Pseudocode

In this section, we provide pseudocodes for both versions of `KLMS`: Algorithm 2.4 with fixed-sized blocks (`Fixed-KLMS`), and Algorithm 2.5 with adaptive-sized blocks (`Adaptive-KLMS`). The algorithms are standalone coding modules that can be applied to different FL frameworks. In the experiments in Section 2.4.4, we used `Adaptive-KLMS` and called it `KLMS` for simplicity. The decoding approach at the server is outlined in Algorithm 2.7.

Algorithm 2.4 `Fixed-KLMS`.

Inputs: post-data $q_{\phi^{(t,c)}}$ and pre-data $p_{\theta^{(t)}}$ distributions, block size S, number of per-block samples K.
Output: selected indices for each block $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$, where $M = \lceil \frac{d}{S} \rceil$ is the number of bloks.

Define $\{q_{\phi_{[m]}^{(t,c)}}\}_{m=1}^{M}$ and $\{p_{\theta_{[m]}^{(t,c)}}\}_{m=1}^{M}$ splitting $q_{\phi^{(t,c)}}$ and $p_{\theta^{(t)}}$ into M distributions on S-size parameters blocks.
for all $m \in \{1,\dots,M\}$
$\quad I \leftarrow [(m-1)S : mS]$.
$\quad$Take K samples from the pre-data distribution: $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta_{[I]}^{(t)}}$.
$\quad\alpha_{[k]} \leftarrow \frac{q_{\phi_{[I]}^{(t,c)}}(\mathbf{y}_{[k]})}{p_{\theta_{[I]}^{(t)}}(\mathbf{y}_{[k]})}\ \forall k \in \{1,\dots,K\}$.
$\quad\pi(k) \leftarrow \frac{\alpha_{[k]}}{\sum_{k'=1}^{K}\alpha_{[k']}}\ \forall k \in \{1,\dots,K\}$.
$\quad$Sample an index $k_{[m]}^{(c)*} \sim \pi(k)$.
end for
Send the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ with $M \cdot \log_2 K$ bits overall for M blocks.

Algorithm 2.5 `Adaptive-KLMS`.

Inputs: post-data $q_{\phi^{(t,c)}}$ and pre-data $p_{\theta^{(t)}}$ distributions, block locations M (a list of start indices of each block), number of per-block samples K, target KL divergence D_{KL}^{target}, the flag `UPDATE` indicating whether the block locations will be updated, the maximum block size allowed `MAX_BLOCK_SIZE`.

Output: selected indices for each block $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$, where the number of blocks M may vary each round.

if `UPDATE`

Construct the sequence of per-coordinate KL-divergence of size d: $\mathbf{D} \leftarrow \left[D_{KL}(q_{\phi_1^{(t,c)}} \| p_{\theta_1^{(t)}}), D_{KL}(q_{\phi_2^{(t,c)}} \| p_{\theta_2^{(t)}}), \ldots, D_{KL}(q_{\phi_d^{(t,c)}} \| p_{\theta_d^{(t)}})\right]$.

Divide $\mathbf{D}$ into subsequences of $\{\mathbf{D}[i_1 = 1 : i_2], \mathbf{D}[i_2 : i_3], \ldots, \mathbf{D}[i_M : i_{M+1} = d]\}$ such that for all $m = 1, \ldots, M$, $\sum_{l=i_m}^{i_{m+1}} \mathbf{D}[l] \approx D_{KL}^{\text{target}}$ or $i_{m+1} - i_m =$ `MAX_BLOCK_SIZE`. Here M, i.e, the number of blocks, may vary each round.

Construct new block locations: $I_m \leftarrow [i_m : i_{m+1}]$ for $m = 1, \ldots, M$.

else

Keep the old block locations I.

end if

Construct per-block post-data $\{q_{\phi_{[I_m]}^{(t,c)}}\}_{m=1}^{M}$ and pre-data $\{p_{\theta_{[I_m]}^{(t)}}\}_{m=1}^{M}$ distributions.

for all $m \in \{1, \ldots, M\}$

Sample $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta_{[I_m]}^{(t)}}$.

$\alpha_{[k]} \leftarrow \frac{q_{\phi_{[I_m]}^{(t,c)}}(\mathbf{y}_{[k]}) \cdot}{p_{\theta_{[I_m]}^{(t)}}(\mathbf{y}_{[k]})} \; \forall k \in \{1, \ldots, K\}$.

$\pi(k) \leftarrow \frac{\alpha_{[k]}}{\sum_{k'=1}^{K} \alpha_{[k']}} \; \forall k \in \{1, \ldots, K\}$.

Sample $k_{[m]}^{(c)*} \sim \pi(k)$.

end for

if `UPDATE`

Return the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ and the new block locations I spending $\approx D_{KL}^{\text{target}} + \log_2($`MAX_BLOCK_SIZE`$)$ bits per block (block sizes are different for each block).

else

Return the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ spending $\approx D_{KL}^{\text{target}}$ bits per block (block sizes are different for each block).

end if

Algorithm 2.6 `Aggregate-Block-Locations`.

Inputs: client block locations $\{I^{(t,c)}\}_{c\in\mathcal{C}_t}$.
Output: new global block locations $I^{(t)}$.

Define empty $I^{(t)}$.
$m_{\max} \leftarrow \max_{c\in\mathcal{C}_t}\left\{\text{length}(I^{(t,c)})\right\}$.
for $m \in \{1, 2, \ldots, m_{\max}\}$
  $\tilde{i}_m \leftarrow 0$.
  $l \leftarrow 0$.
  for $c \in \mathcal{C}_t$
    if $\text{length}(I^{(t,c)}) \geq m$
      $\tilde{i}_m \leftarrow \tilde{i}_m + I^{(t,c)}_{i_m}$.
      $l \leftarrow l + 1$.
    end if
  end for
  $\bar{i}_m \leftarrow \lceil \tilde{i}_m / l \rceil$.
  Add $\bar{i}_m$ to $I^{(t)}$.
end for
Return $I^{(t)}$.

Algorithm 2.7 `KLMS`-Decoder.

Inputs: pre-data $p_{\theta^{(t)}}$ distribution, block locations I of M blocks, number of per-block samples K, selected indices for each block $\{k^{(c)*}_{[m]}\}_{m=1}^{M}$, where $M = \lceil \frac{d}{S} \rceil$ is the number of blocks.
Output: The selected samples $\{\mathbf{y}^*_{[m]}\}_{m=1}^{M}$ for each block.

Define $\{p_{\theta^{(t)}_{[I_m]}}\}_{m=1}^{M}$ splitting $p_{\theta^{(t)}}$ into M distributions with block locations in I.
for all $m \in \{1, \ldots, M\}$
  Take K samples from the pre-data distribution: $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta^{(t)}_{[I_m]}}$.
  Recover $\mathbf{y}^*_{[m]} \leftarrow \mathbf{y}_{k^{(c)}_{[m]}}$.
end for
Return the selected samples $\{\mathbf{y}^*_{[m]}\}_{m=1}^{M}$ for each block.

Pseudocode for Applications KLMS

Algorithm 2.8 FedPM-KLMS.

Hyperparameters: thresholds to update block locations $\bar{D}_{KL}^{\max}$ and $\bar{D}_{KL}^{\min}$, maximum block size MAX_BLOCK_SIZE.

Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.

Output: random SEED and binary mask parameters $\boldsymbol{m}^{(T)}$.

At the server, initialize a random network with weight vector $\boldsymbol{w}^{\text{init}} \in \mathbb{R}^d$ using a random SEED, and broadcast it to the clients; initialize the random score vector $\boldsymbol{s}^{(0,g)} \in \mathbb{R}^d$, and compute $\theta^{(0,g)} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{(0,g)})$, Beta priors $\boldsymbol{\alpha}^{(0)} = \boldsymbol{\beta}^{(0)} = \boldsymbol{\lambda}_0$; initialize UPDATE←TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.

for $t = 1, \ldots, T$

Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.

On Client Nodes:

for $c \in \mathcal{C}_t$

Compute $\phi^{(t,c)}$ as in FedPM.

if UPDATE

$\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ Adaptive-KLMS$(\text{Bern}(\theta^{(t,g)}), \text{Bern}(\phi^{(t,c)}), I^{(t)}, D_{KL}^{\text{target}})$ // Alg. 2.5.

$M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.

else

$\{k_{[i]}^*\}_{i=1}^M \leftarrow$ Adaptive-KLMS$(\text{Bern}(\theta^{(t,g)}), \text{Bern}(\phi^{(t,c)}), I^{(t)}, D_{KL}^{\text{target}})$ // Alg. 2.5.

end if

Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(\text{Bern}(\phi_{[I_m]}^{(t,c)}) \| \text{Bern}(\theta_{[I_m]}^{(t,g)}))$ with 32 bits to the server.

if UPDATE

Send $I^{(t,c)}$ with $M \cdot \log_2($MAX_BLOCK_SIZE$)$ bits.

end if

end for

On the Server Node:

Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.

Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.

if UPDATE

$I^{(t)} \leftarrow$ Aggregate-Block-Locations $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 2.6.

UPDATE = False.

else

$I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.

if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** UPDATE = True **else** UPDATE = False.

end if

for $c \in \mathcal{C}_t$

$\{\hat{\boldsymbol{m}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ KLMS-Decoder$(\text{Bern}(\theta^{(t)}), I^{(t,c)}, K)$ // See Algorithm 2.7.

end for

$\theta^{(t)} = \text{BayesAgg}\left(\{\hat{\boldsymbol{m}}^{(t,c)}\}_{c\in\mathcal{C}_t}, t\right)$ // See Algorithm 2.3.

Broadcast UPDATE, $I^{(t)}$ and $\theta^{(t)}$ to the clients.

end for

Sample $\boldsymbol{m}^{\text{final}} \sim \text{Bern}(\theta^{(T)})$ and return the final model $\dot{\boldsymbol{w}}^{\text{final}} \leftarrow \boldsymbol{m}^{\text{final}} \odot \boldsymbol{w}^{\text{init}}$.

Algorithm 2.9 `QSGD-KLMS`.

Hyperparameters: server learning rate η_S, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: Final model $\boldsymbol{w}^{(T)}$.

At the server, initialize a random network parameters $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`$\leftarrow$TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.
for $t = 1, \ldots, T$
  Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$
    Receive the empirical frequency from the previous round $p_{\theta^{(t)}}$ from the server.
    Compute $\mathbf{v}^{(t,c)}$ as in `QSGD`.
    Compute the local post-data distribution $q_{\phi^{(t,c)}}$ with $\mathbf{v}^{(t,c)}$ using $p_{\texttt{QSGD}}(\cdot)$ in Eq. (2.24).
    if `UPDATE`
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.
      $M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t)}})$ with 32 bits to the server.
    if UPDATE
      Send $I^{(c)}$ with $M \cdot \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits.
    end if
  end for
  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE`
    $I^{(t,c)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 2.6.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$
    $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS-Decoder`$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 2.7.
    Construct the empirical frequency $p_{\theta^{(t+1)}}$ from $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M$.
  end for
  Compute $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{\mathbf{v}}^{(t,c)}$.
  Broadcast `UPDATE`, $I^{(t)}$, $\boldsymbol{w}^{(t)}$, and $p_{\theta^{(t)}}$ to the clients.
end for

Algorithm 2.10 `SignSGD-KLMS`.

Hyperparameters: server learning rate η_S, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: Final model $\boldsymbol{w}^{(T)}$.

At the server, initialize a random network parameters $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`$\leftarrow$TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.
for $t = 1, \ldots, T$
  Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$
    Compute $\mathbf{v}^{(t,c)}$ as in other standard FL frameworks such as `QSGD`.
    Compute the local post-data distribution $q_{\phi^{(t,c)}}$ with $\mathbf{v}^{(t,c)}$ using $p_{\texttt{SignSGD}}(\cdot)$ in Eq. (2.25).
    $p_{\theta^{(t)}} \leftarrow$ Unif(0.5) over $\{-1, 1\}$.
    if `UPDATE`
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.
      $M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t,g)}}))$ with 32 bits to the server.
    if UPDATE
      Send $I^{(t,c)}$ with $M \cdot \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits.
    end if
  end for
  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE`
    $I^{(t)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 2.6.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$
    $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS-Decoder`$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 2.7.
  end for
  Compute $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{\mathbf{v}}^{(t,c)}$.
  Broadcast `UPDATE`, $I^{(t)}$ and $\boldsymbol{w}^{(t)}$ to the clients.
end for

Algorithm 2.11 `SGLD-KLMS`.

Hyper-parameters: server learning rate η_S, minibatch size B, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.

Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.

Output: samples $\left\{\theta^{(t)}\right\}_{t=1}^{T}$.

At the server, initialize a random network with weight vector $\theta^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`$\leftarrow$TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.

for $t = 1, \ldots, T$

Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.

On Client Nodes:

for $c \in \mathcal{C}_t$

Receive $\theta^{(t-1)}$ from the server and set $\phi^{(t,c)} \leftarrow \theta^{(t-1)}$.

Compute a stochastic gradient of the potential $H(\phi^{(t,c)})$ as in `QLSD`.

Set $p_{\theta^{(t)}} \leftarrow \mathcal{N}\left(0, \sqrt{\frac{2}{\gamma C^2}}\boldsymbol{I}_d\right)$.

Set $q_{\phi^{(t,c)}} \leftarrow \mathcal{N}\left(H(\phi^{(t,c)}), \sqrt{\frac{2}{\gamma C^2}}\boldsymbol{I}_d\right)$.

if `UPDATE`

$\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.

$M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.

else

$\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 2.5.

end if

Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^{M} D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t,g)}}))$ with 32 bits to the server.

if UPDATE

Send $I^{(t,c)}$ with $M \cdot \log_2($`MAX_BLOCK_SIZE`$)$ bits.

end if

end for

On the Server Node:

Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.

Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.

if `UPDATE`

$I^{(t)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 2.6.

`UPDATE` = `False`.

else

$I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.

if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.

end if

for $c \in \mathcal{C}_t$

$\{\hat{H}(\phi_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS-Decoder`$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 2.7.

end for

Compute $\theta^{(t)} = \theta^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{H}(\phi^{(t,c)})$.

Broadcast `UPDATE`, $I^{(t)}$ and $\theta^{(t)}$ to the clients.

end for

2.7.2 Proofs

Proof of Eq. 2.15

We now provide proof of the upper bound on the estimation error in Eq. 2.15. Recall that our true mean is $\bar{\boldsymbol{\theta}}^{g,t} = \frac{1}{K}\sum_{k\in\mathcal{K}_t}\boldsymbol{\theta}^{k,t}$, whereas our estimate is $\bar{\boldsymbol{\theta}}^{g,t} = \frac{1}{K}\sum_{k\in\mathcal{K}_t}\boldsymbol{m}^{k,t}$, where $\boldsymbol{m}^{k,t} \sim \text{Bern}(\boldsymbol{\theta}^{k,t})$. Then we can compute the error as

$$\mathbb{E}_{\boldsymbol{M}^{k,t}\sim\text{Bern}(\boldsymbol{\theta}^{k,t})\forall k\in\mathcal{K}_t}\left[||\widehat{\bar{\boldsymbol{\theta}}}^{g,t} - \bar{\boldsymbol{\theta}}^{g,t}||_2^2\right] = \sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\ \forall k\in\mathcal{K}_t}\left[\left(\widehat{\bar{\theta}}_i^{g,t} - \bar{\theta}_i^{g,t}\right)^2\right] \tag{2.41}$$

$$= \sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in\mathcal{K}_t}\left[\left(\frac{1}{K}\sum_{k\in\mathcal{K}_t}(M_i^{k,t} - \theta_i^{k,t})\right)^2\right] \tag{2.42}$$

$$= \frac{1}{K^2}\sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in\mathcal{K}_t}\left[\left(\sum_{k\in\mathcal{K}_t}(M_i^{k,t} - \theta_i^{k,t})\right)^2\right] \tag{2.43}$$

$$= \frac{1}{K^2}\sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in\mathcal{K}_t}\left[\sum_{k\in\mathcal{K}_t}\left(M_i^{k,t} - \theta_i^{k,t}\right)^2\right] \tag{2.44}$$

$$= \frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in\mathcal{K}_t}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})}\left[(M_i^{k,t} - \theta_i^{k,t})^2\right] \tag{2.45}$$

$$= \frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in\mathcal{K}_t}\left(\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})}[(M_i^{k,t})^2] - (\theta_i^{k,t})^2\right) \tag{2.46}$$

$$= \frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in\mathcal{K}_t}\left(\theta_i^{k,t} - (\theta_i^{k,t})^2\right) \tag{2.47}$$

$$\leq \frac{d}{4K}. \tag{2.48}$$

From (2.41) to (2.42), we use the definition of $\widehat{\bar{\theta}}_i^{g,t} = \frac{1}{K}\sum_{k=1}^{K} m_i^{k,t}$ and $\bar{\theta}_i^{g,t} = \frac{1}{K}\sum_{k=1}^{K}\theta_i^{k,t}$. From (2.43) to (2.44), we use the fact that $\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\ \forall k\in\mathcal{K}_t}[M_i^{k,t} - \theta_i^{k,t}] = 0$; and $M_i^{k,t} - \theta_i^{k,t}$ and $M_i^{l,t} - \theta_i^{l,t}$ are independent for $l \neq k \in [K]$. Finally, the inequality in (2.48) follows from $\theta_i^{k,t} \in [0,1]$ for all $k \in [K]$.

Proof of Theorem 5

In this section, we provide the proof for Theorem 5. But before that, we first define the formal problem statement, introduce some new notation, and give another theorem (Theorem 18) that will be required for the proof of Theorem 5.

We consider a scenario where N distributed nodes and a centralized server share a prior distribution p_θ over a set $\mathcal{X}$ equipped with some sigma algebra. Each node n also holds a posterior distribution $q_{\phi^{(n)}}$ over the same set. The server wants to estimate $\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}}\forall n\in[N]}[\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)})]$,

where $f(\cdot) : \mathcal{X} \to \mathbb{R}$ is a measurable function. In order to minimize the cost of communication from the nodes to the centralized server, each node n and the centralized server take $K^{(n)}$ samples from the prior distribution $\mathbf{y}_{[1]}^{(n)}, \ldots, \mathbf{y}_{[K^{(n)}]}^{(n)} \sim p_\theta$. Then client n performs the following steps:

1. Define a new probability distribution over the indices $k = 1, \ldots, K^{(n)}$:

$$\pi^{(n)}(k) = \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})} \tag{2.49}$$

   and over the samples $\mathbf{y}_{[1]}^{(n)}, \ldots, \mathbf{y}_{[K^{(n)}]}^{(n)}$:

$$\tilde{q}_{\pi^{(n)}}(\mathbf{y}) = \sum_{k=1}^{K^{(n)}} \pi^{(n)}(k) \cdot \mathbf{1}(\mathbf{y}_{[k]}^{(n)} = \mathbf{y}). \tag{2.50}$$

2. Sample $k^{(n)*} \sim \pi^{(n)}$.

3. Communicate $k^{(n)*}$ to the centralized server with $\log K^{(n)}$ bits.

Then, the centralized server recovers the sample $\mathbf{y}_{[k^{(n)*}]}^{(n)}$ that it generated in the beginning. (Note that $\mathbf{y}_{[k^{(n)*}]}^{(n)}$ is actually a sample from $\tilde{q}_{\pi^{(n)}}$.) Finally, the server aggregates these samples $\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{y}_{k^{(n)*}}^{(n)})$ which is an estimate of

$$\mathbb{E}_{Y^{(n)} \sim \tilde{q}_{\pi^{(n)}} \forall n \in [N]}[\frac{1}{N}\sum_{m=1}^{N} f(Y^{(m)})]. \tag{2.51}$$

We want to find a relation between the number of samples $K^{(1)}, \ldots, K^{(N)}$ (or the number of bits $\log K^{(1)}, \ldots, \log K^{(N)}$) and the error in the estimate, $|\mathbb{E}_{Y^{(n)} \sim \tilde{q}_{\pi^{(n)}} \forall n \in [N]}[\frac{1}{N}\sum_{m=1}^{N} f(Y^{(m)})] - \mathbb{E}_{X^{(n)} \sim q_{\phi^{(n)}} \forall n \in [N]}[\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)})]|$. In our proofs, we closely follow the methodology in Theorems 1.1. and 1.2. in [58]. In Theorem 18, we use the probability density of $q_{\phi^{(n)}}$ with respect to p_θ for each node n and denote it by $\rho_n = \frac{dq_{\phi^{(n)}}}{dp_\theta}$. We refer to the following definitions often:

$$I(f) = \int_{\mathbf{x}^{(1)}} \cdots \int_{\mathbf{x}^{(N)}} \left(\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}^{(n)})\right) \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)}), \tag{2.52}$$

$$I_K(f) = \frac{1}{\prod_{n=1}^{N} K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \left(\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{y}_{[k^{(n)}]}^{(n)})\right) \prod_{n=1}^{N} \rho_n(\mathbf{y}_{[k^{(n)}]}^{(n)}), \tag{2.53}$$

and

$$J_K(f) = \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \left(\frac{1}{N} \sum_{n=1}^{N} f(\mathbf{y}_{[k^{(n)}]}^{(n)}) \right) \prod_{n=1}^{N} \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k^{(n)}]}^{(n)})/p_\theta(\mathbf{y}_{[k^{(n)}]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})}. \tag{2.54}$$

Notice that $I(f)$ corresponds to the target value the centralized server wants to estimate, $J_K(f)$ is the estimate from the proposed approach, and $I_K(f)$ is a value that will be useful in the proof and that satisfies $\mathbb{E}[I_K(f)] = I(f)$.

Theorem 18. *Let p_θ and $q_{\phi^{(n)}}$ for $n = 1, \ldots, N$ be probability distributions over a set $\mathcal{X}$ equipped with some sigma-algebra. Let $X^{(n)}$ be an $\mathcal{X}$-valued random variable with law $q_{\phi^{(n)}}$. Let $r \geq 0$ and $\tilde{q}_{\pi^{(n)}}$ for $n = 1, \ldots, N$ be discrete distributions each constructed by $K^{(n)} = \exp\{(D_{KL}(q_{\phi^{(n)}} \| p_\theta) + r)\}$ samples $\{\mathbf{y}_{[k^{(n)}]}^{(n)}\}_{k^{(n)}=1}^{K^{(n)}}$ from p_θ defining $\tilde{q}_{\pi^{(n)}}(\mathbf{y}) = \sum_{k=1}^{K^{(n)}} \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})} \cdot \mathbf{1}(\mathbf{y}_{[k]}^{(n)} = \mathbf{y})$. Furthermore, for $f(\cdot)$ defined above, let $||f||_{\mathbf{q}_\phi} = \sqrt{\mathbb{E}_{X^{(n)} \sim q_{\phi^{(n)}} \forall n \in [N]}[(\frac{1}{N} \sum_{m=1}^{N} f(X^{(m)}))^2]}$ be its 2-norm under $\mathbf{q}_\phi = q_{\phi^{(1)}}, \ldots, q_{\phi^{(N)}}$. Then,*

$$\mathbb{E}|I_K(f) - I(f)| \leq ||f||_{\mathbf{q}_\phi} \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) > D_{KL}(q_{\phi^{(n)}} || p_\theta) + r/2\right)} \right). \tag{2.55}$$

Conversely, let $\mathbf{1}$ denote the function from $\mathcal{X}$ into $\mathbb{R}$ that is identically equal to 1. If for $n = 1, \ldots, N$, $K^{(n)} = \exp\{(D_{KL}(q_{\phi^{(n)}} || p_\theta) - r)\}$ for some $r \geq 0$, then for any $\delta \in (0, 1)$,

$$\mathbb{P}(I_K(\mathbf{1}) \geq 1 - \delta) \leq e^{-Nr/2} + \frac{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) \leq D_{KL}(q_{\phi^{(n)}} || p_\theta) - r/2\right)}{1 - \delta}. \tag{2.56}$$

Proof. Let $L^{(n)} = D_{KL}(q_{\phi^{(n)}} || p_\theta), \forall n \in [N]$. Suppose that $K^{(n)} = e^{L^{(n)}+r}$ and $a^{(n)} = e^{L^{(n)}+r/2}$. Let $h(z) = f(z)$ if $\rho_n(z) \leq a^{(n)}$ and 0 otherwise $\forall n \in [N]$. We first make the following assumption:

$$\begin{aligned} \mathbb{E}[|\frac{1}{N} \sum_{n \in Q \subseteq [N]} f(X^{(n)})|; \forall n \in Q \subseteq [N], \rho_n(X^{(n)}) > a^{(n)}] \leq \\ \mathbb{E}[|\frac{1}{N} \sum_{n \in [N]} f(X^{(n)})|; \forall n \in [N], \rho_n(X^{(n)}) > a^{(n)}]. \end{aligned} \tag{2.57}$$

This is indeed a reasonable assumption. To see this, following [58], we note that $\log \rho_n(Z)$ is concentrated around its expected value, which is $L^{(n)} = D_{KL}(q_{\phi^{(n)}} || p_\theta)$, in many scenarios. Therefore, for small t (and t is indeed negligibly small in our experiments), the events $\mathbf{1}\{\forall n \in Q \subseteq [N], \rho_n(X^{(n)}) > a^{(n)}\}$ occur with the approximately same frequency for each set $Q \subseteq [N]$ since the likelihood of event $\mathbf{1}\{\rho_n(X^{(n)}) > a^{(n)}\}$ is close to being uniform. Consider also that

$|\frac{1}{N}\sum_{n\in Q\subseteq[N]} f(X^{(n)})| \leq |\frac{1}{N}\sum_{n\in[N]} f(X^{(n)})|$ holds when $f(X^n)$'s have the same signs per coordinate for each $n = 1, \ldots, N$, which is a realistic assumption given that the clients are assumed to be able to train a joint model and hence should not have opposite signs in the updates very often. With these two observations, we argue that the assumption in Eq. (2.57) is indeed reasonable for many scenarios, including FL.

Now, going back to the proof, from triangle inequality, we have,

$$|I_K(f) - I(f)| \leq |I_K(f) - I_K(h)| + |I_K(h) - I(h)| + |I(h) - I(f)|. \tag{2.58}$$

First, note that by Cauchy-Schwarz inequality and by the assumption in Eq. (2.57), we have

$$\begin{aligned}
|I(h) - I(f)| &= \sum_{Q\subseteq[N]} \mathbb{E}[|\frac{1}{N}\sum_{m\in Q} f(X^{(m)})|; \forall n \in Q, \rho_n(X^{(n)}) > a^{(n)}]\cdot \\
&\quad \cdot \mathbb{P}(\forall n \in Q, \rho_n(X^{(n)}) > a^{(n)}) && (2.59)\\
&\leq \mathbb{E}[|\frac{1}{N}\sum_{m\in[N]} f(X^{(m)})|; \forall n \in [N], \rho_n(X^{(n)}) > a^{(n)}] \sum_{Q\subseteq[N]} \mathbb{P}(\forall n \in Q, \rho_n(X^{(n)}) > a^{(n)}) && (2.60)\\
&= \mathbb{E}[|\frac{1}{N}\sum_{m\in[N]} f(X^{(m)})|; \forall n \in [N], \rho_n(X^{(n)}) > a^{(n)}] && (2.61)\\
&= \int_{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)}} |\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}^{(n)})| \cdot 1\{\forall n \in [N], \rho_n(\mathbf{x}^{(n)}) > a^{(n)}\} \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)}) && (2.62)\\
&\leq \sqrt{\int_{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)}} |\frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}^{(m)})|^2 \cdot \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)})}\cdot \\
&\quad \cdot \sqrt{\int_{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)}} 1\{\forall n \in [N], \rho_n(\mathbf{x}^{(n)}) > a^{(n)}\} \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)})} && (2.63)\\
&= \sqrt{\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}}, \forall n\in[N]}[(\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)}))^2]} \cdot \sqrt{\mathbb{P}(\forall n \in [N], \rho_n(X^{(n)}) > a^{(n)})} && (2.64)\\
&= ||f||_{\mathbf{q}_\phi} \cdot \sqrt{\mathbb{P}(\forall n \in [N], \rho_n(X^{(n)}) > a^{(n)})}. && (2.65)
\end{aligned}$$

Similarly,

$$\mathbb{E}|I_K(f) - I_K(h)| = \mathbb{E}\left|\frac{1}{\prod_{n=1}^N K^{(n)}}\sum_{k^{(1)}=1}^{K^{(1)}}\cdots\sum_{k^{(N)}=1}^{K^{(N)}}\frac{1}{N}(\sum_{m=1}^N f(Y_{[k^{(m)}]}^{(m)}) - h(Y_{[k^{(m)}]}^{(m)}))\prod_{n=1}^N \rho_n(Y_{[k^{(n)}]}^{(n)})\right| \tag{2.66}$$

$$\leq \mathbb{E}\left|\frac{1}{N}(\sum_{m=1}^N f(Y_{[k^{(m)}]}^{(m)}) - h(Y_{[k^{(m)}]}^{(m)}))\prod_{n=1}^N \rho_n(X^{(n)})\right| \tag{2.67}$$

$$= \mathbb{E}[|\frac{1}{N}\sum_{m=1}^N f(X^{(m)})|; \forall n \in [N], \rho_n(X^{(n)}) > a^{(n)}] \tag{2.68}$$

$$\leq ||f||_{\mathbf{q}_\phi} \cdot \sqrt{\mathbb{P}(\forall n \in [N], \rho_n(X^{(n)}) > a^{(n)})}. \tag{2.69}$$

From Eq. (2.68) to Eq. (2.69), we follow the same steps in Eq. (2.61)-Eq. (2.65). Finally, note that

$$\mathbb{E}|I_K(h) - I(h)| \leq \sqrt{Var(I_K(h))} \tag{2.70}$$

$$= \sqrt{\frac{1}{\prod_{n=1}^N K^{(n)}} Var\left(\frac{1}{N}\sum_{m=1}^N h(Y_{[1]}^{(m)}) \cdot \prod_{n=1}^N \rho_n(Y_{[1]}^{(n)})\right)} \tag{2.71}$$

$$\leq \sqrt{\frac{1}{\prod_{n=1}^N K^{(n)}} \mathbb{E}\left[(\frac{1}{N}\sum_{m=1}^N h(Y_{[1]}^{(n)}))^2 \prod_{n=1}^N (\rho_n(Y_{[1]}^{(n)}))^2\right]} \tag{2.72}$$

$$\leq \sqrt{\frac{\prod_{n=1}^N a^{(n)}}{\prod_{n=1}^N K^{(n)}} \mathbb{E}\left[(\frac{1}{N}\sum_{m=1}^N f(Y_{[1]}^{(m)}))^2 \prod_{n=1}^N \rho_n(Y_{[1]}^{(n)})\right]} \tag{2.73}$$

$$= ||f||_{\mathbf{q}_\phi} \prod_{n=1}^N \left(\frac{a^{(n)}}{K^{(n)}}\right)^{1/2}. \tag{2.74}$$

Combining the upper bounds above, we get

$$\mathbb{E}\left[|I_K(f) - I(f)|\right] \leq ||f||_{\mathbf{q}_\phi}\left(\prod_{n=1}^N \left(\frac{a^{(n)}}{K^{(n)}}\right)^{1/2} + 2\sqrt{\prod_{n=1}^N \mathbb{P}\left(\log \rho_n(X^{(n)}) > \log a^{(n)}\right)}\right) \tag{2.75}$$

$$= ||f||_{\mathbf{q}_\phi}\left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^N \mathbb{P}\left(\log \rho_n(X^{(n)}) > L^{(n)} + r/2\right)}\right) \tag{2.76}$$

$$= ||f||_{\mathbf{q}_\phi}\left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^N \mathbb{P}\left(\log \rho_n(X^{(n)}) > D_{KL}(q_{\phi^{(n)}}||p) + r/2\right)}\right). \tag{2.77}$$

This completes the proof of the first part of the theorem.

For the converse part, suppose $K^{(n)} = e^{L^{(n)}-r}$ and $a^{(n)} = e^{L^{(n)}-r/2}\ \forall n \in [N]$. Then,

$$\mathbb{P}(I_K(\mathbf{1}) \geq 1-\delta) = \mathbb{P}\left(\frac{1}{\prod_{n=1}^N K^{(n)}} \sum_{k_1=1}^{K_1} \cdots \sum_{k_N=1}^{K_N} \prod_{n=1}^N \rho_n(Y^{(n)}_{[k^{(n)}]}) \geq 1-\delta\right) \tag{2.78}$$

$$\begin{aligned}\leq &\mathbb{P}\left(\max_{1\leq k\leq K^{(n)}} \rho_n(Y^{(n)}_{[k]}) > a^{(n)}, \forall n \in [N]\right) \\ &+ \mathbb{P}\left(\frac{1}{\prod_{n=1}^N K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^N \rho_n(Y^{(n)}_{[k^{(n)}]}) 1\{\forall n \in [N], \rho_n(Y^{(n)}_{[k^{(n)}]}) \leq a^{(n)}\} \geq 1-\delta\right)\end{aligned} \tag{2.79}$$

$$\begin{aligned}\leq &\sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \mathbb{P}\left(\rho_n(Y^{(n)}_{[k^{(n)}]}) > a^{(n)}, \forall n \in [N]\right) \\ &+ \frac{1}{1-\delta}\mathbb{E}\left[\frac{1}{\prod_{n=1}^N K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^N \rho_n(Y^{(n)}_{[k^{(n)}]}) 1\{\forall n \in [N], \rho_n(Y^{(n)}_{[k^{(n)}]}) \leq a^{(n)}\}\right]\end{aligned} \tag{2.80}$$

$$\leq \frac{1}{\prod_{n=1}^N a^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^N \mathbb{E}\left[\rho_n(Y^{(n)}_{[k^{(n)}]})\right] + \frac{1-\prod_{n=1}^N \mathbb{P}\left(\rho_n(Z) \geq a^{(n)}\right)}{1-\delta} \tag{2.81}$$

$$= \prod_{n=1}^N \frac{K^{(n)}}{a^{(n)}} + \frac{\prod_{n=1}^N \mathbb{P}\left(\rho_n(Z) \leq a^{(n)}\right)}{1-\delta} \tag{2.82}$$

$$= e^{-Nr/2} + \frac{\prod_{n=1}^N \mathbb{P}\left(\log \rho_n(X^{(n)}) \leq D_{KL}(q_{\phi^{(n)}}||p_\theta) - r/2\right)}{1-\delta}, \tag{2.83}$$

where from (2.78) to (2.80) and (2.79) to (2.80), we use Markov's inequality. This completes the proof of the second inequality in the theorem statement. □

Now, we restate Theorem 5 below and provide the proof afterward.

Theorem 19 (Theorem 5)**.** *Let all notations be as in Theorem 18 and let $J_K(f)$ be the estimate defined in (2.54). Suppose that $K^{(n)} = \exp\{(L^{(n)} + r)\}$ for some $r \geq 0$. Let*

$$\epsilon = \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^N \mathbb{P}(\log \rho_n(X^{(n)}) > L^{(n)} + r/2)}\right)^{1/2}. \tag{2.84}$$

Then

$$\mathbb{P}\left(|J_K(f) - I(f)| \geq \frac{2||f||_{\mathbf{q}_\phi}\epsilon}{1-\epsilon}\right) \leq 2\epsilon. \tag{2.85}$$

Proof. Suppose that $K^{(n)} = e^{L^{(n)}+r}$ and $a^{(n)} = e^{L^{(n)}+r/2}$ $\forall n \in [N]$. Let

$$b = \sqrt{\prod_{n=1}^{N} \frac{a^{(n)}}{K^{(n)}}} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\rho_n(X^{(n)}) > a^{(n)}\right)}. \tag{2.86}$$

Then, by Theorem 18, for any $\epsilon, \delta \in (0, 1)$,

$$\mathbb{P}\left(|I_K(1) - 1| \geq \epsilon\right) \leq \frac{b}{\epsilon} \tag{2.87}$$

and

$$\mathbb{P}\left(|I_K(f) - I(f)| \geq \delta\right) \leq \frac{||f||_{\mathbf{q}_\phi} b}{\delta}. \tag{2.88}$$

Now, if $|I_K(f) - I(f)| < \delta$ and $|I_K(1) - 1| < \epsilon$, then

$$|J_K(f) - I(f)| = \left|\frac{I_K(f)}{I_K(1)} - I(f)\right| \tag{2.89}$$

$$\leq \frac{|I_K(f) - I(f)| + |I(f)||1 - I_K(1)|}{I_K(1)} \tag{2.90}$$

$$< \frac{\delta + |I(f)|\epsilon}{1 - \epsilon}. \tag{2.91}$$

Taking $\epsilon = \sqrt{b}$ and $\delta = ||f||_{\mathbf{q}_\phi}\epsilon$ completes the proof of the first inequality in the theorem statement. Note that if ϵ is bigger than 1, the bound is true anyway.

This completes the proof of the theorem. □

Proof of Theorem 12

To prove Theorem 12, we need to show that we can translate the average distortion d_{avg} requirement into a constraint on the empirical distribution $\hat{Q}_{s^n\hat{h}^n}(d,h) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{(s_i,h_i)=(s,h)}$, where $\mathbf{1}_{\mathcal{I}} = 1$ if the condition $\mathcal{I}$ is true, and 0 otherwise. We can write

$$\begin{aligned}
d_{avg}(Q^n,\hat{Q}^n) &= \frac{1}{n}\sum_{i=1}^{n} d(Q_i,\hat{Q}_i) \\
&= \mathbb{E}_{C,S}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_C(\hat{Q}_i) - \mathcal{L}_C(Q_i)\right] \\
&\overset{(a)}{=} \mathbb{E}_{C,S}\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{h\in\mathcal{H}}\mathcal{L}_C(h)(\mathbf{1}_{(S,\hat{h}_i)=(S,h)} - Q_{h|S})\right] \\
&\overset{(b)}{=} \mathbb{E}_{C,S}\left[\sum_{h\in\mathcal{H}}\mathcal{L}_C(h)\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{(S,\hat{h}_i)=(S,h)} - Q_{h|S}\right)\right] \\
&= \mathbb{E}_{C,S}\left[\sum_{h\in\mathcal{H}}\mathcal{L}_C(h)\left(\hat{Q}_{s^{n'}\hat{h}^{n'}}(d,h) - Q_{h|S}\right)\right] \\
&= \mathbb{E}_{C,S}\left[\mathcal{L}(\hat{Q}_{s^{n'}h^{n'}}) - \mathcal{L}(Q)\right] \\
&= d_{sem}(Q,\hat{Q}_{s^{n'}\hat{h}^{n'}}),
\end{aligned}$$

where the indicator function in (a) depends on the chosen $(2^{nR},n)$ coding scheme, and (b) comes from the fact that Alice's sampling scheme does not depend on the model index i. We then showed that the average distortion requirement translates into a distortion between the empirical distribution $\hat{Q}_{s^n\hat{h}^n}$, and the target Q. Given this, Theorem 1 in [140] applies, and our Theorem 12 follows.

Proof of Lemma 15

Proof. Theorem 12 provides the minimum achievable rate to satisfy $d_{avg} \leq \epsilon$, which is provided by an optimized distribution $Q^*_{S,H}$ that minimizes the mutual information $I(S;H)$, and satisfies $d_{avg} \leq \epsilon$. Now, we impose strong coordination between Alice and Bob, by using as target joint distribution $Q^*_{S^n,H^n} = \prod_{i=1}^{n} Q^*_{S,H}$, and by Theorem 10 of [71] the same rate $I(S;H)$ of empirical coordination can be achieved, as long as enough common randomness is available. Now, given that Bob's distribution $\hat{Q}_{S^n,\hat{H}^n}$ converges in total variation to $Q^*_{S^n,H^n}$, so it happens for the marginal, i.e., $\hat{Q}_{S^n,\hat{H}^n}(s_i,\hat{h}_i) \xrightarrow{\text{TV}} Q^*_{S,H}$. However, by construction, $Q^*_{S,H}$ satisfies the distortion constraint symbol-wise, and so satisfies d_{max}. □

Proof of Theorem 16

To prove Theorem 16, we observe that Alice now needs to convey her probability distribution $Q^n_{H^n|S^n}$ to Bob using $P^n_H = (P_{H^n})^n$ to code it. If we indicate with $\mathbb{E}_{C,S}[K]$ the average number

of bits spent to convey model belief Q using distribution P, Corollary 3.4 in [235] provides the single-shot bounds

$$\mathcal{R}(Q,P) \leq \mathbb{E}_{C,S}\left[K\right] \leq \mathcal{R}(Q,P) + \log\left(\mathcal{R}(Q,P)+1\right) + 4,$$

where $\mathcal{R}(Q,P) = \mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(Q||P)\right]$.

We now translate the results to the n-length sequence, and compute its limit as n grows indefinitely.

$$\begin{aligned}
\mathcal{R}(Q^n,P^n) &= \mathbb{E}_{C^n,S^n}\left[D_{\mathrm{KL}}(Q^n||P^n)\right] \\
&= \mathbb{E}_{C^n}\left[\mathbb{E}_{S^n|C^n}\left[\mathbb{E}_{H^n|S^n}\log\frac{Q^n_{H^n|S^n}}{P^n_{H^n}}\right]\right] \\
&\overset{(a)}{=} \mathbb{E}_{C^n}\left[\mathbb{E}_{S^n|C^n}\left[\mathbb{E}_{H^n|S^n}\log\frac{\prod_{i=1}^n Q_{H|S_i}}{(P_H)^n}\right]\right] \\
&\overset{(b)}{=} \mathbb{E}_{C^n}\left[\mathbb{E}_{S^n|C^n}\left[\mathbb{E}_{H^n|S^n}\log\frac{(Q_{H|S})^n}{(P_H)^n}\right]\right] \\
&= n\mathcal{R}(Q,P),
\end{aligned}$$

where (a) is because samples are independent given the nature of the problem, and (b) is because, given the dataset realization, they are also identically distributed by our assumption. Consequently, we can upper bound the total number of bits with

$$\mathbb{E}_{C,S}\left[K\right] \leq n\mathcal{R}(Q,P) + \log\left(n\mathcal{R}(Q,P)+1\right) + 4,$$

obtaining an average rate of

$$\lim_{n\to\infty}\frac{\mathbb{E}_{C,S}\left[K\right]}{n} \to \mathcal{R}(Q,P).$$

Proof of Lemma 17

We now prove the result in Lemma 17. Let Q be the distribution at the sender, and $\hat{Q}$ the one at the receiver, i.e., the distribution $\hat{Q} = Q^*_{S,H}$ in the proof of Lemma 15. We first bound, for each

$C \in \mathcal{C}$, d_{max} by noticing that, for each $i \in \{1, \dots, n\}$, the following holds

$$
\begin{aligned}
& d_{sem}(Q_i^n, \hat{Q}_i^n) = \\
& \overset{(a)}{=} \mathbb{E}_{C,S}\left[\mathcal{L}_C(\hat{Q}) - \mathcal{L}_C(Q)\right] \\
& = \mathbb{E}_{C,S}\left[\sum_{h \in \mathcal{H}} \mathbb{E}_{z \sim C}\left[\ell(h,z)\right]\left(\hat{Q}_{\hat{h}|S} - Q_{h|S}\right)\right] \\
& \leq L_{\max} \cdot \mathbb{E}_{C,S}\left[\sum_{h \in \mathcal{H}} \hat{Q}_{\hat{h}|S} - Q_{h|S}\right] \\
& \leq L_{\max} \cdot \mathbb{E}_{C,S}\left[\mathrm{TV}(\hat{Q}, Q)\right] \\
& \overset{(b)}{\leq} L_{\max} \cdot \mathbb{E}_{C,S}\Big[\min\Big\{\sqrt{\frac{1}{2} D_{\mathrm{KL}}(\hat{Q}\|Q)}, \sqrt{1 - e^{-D_{\mathrm{KL}}(\hat{Q}\|Q)}}\Big\}\Big] \\
& \overset{(c)}{\leq} L_{\max} \cdot \min\Big\{\sqrt{\frac{1}{2}\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(\hat{Q}\|Q)\right]}, \sqrt{1 - e^{-\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(\hat{Q}\|Q)\right]}}\Big\}
\end{aligned}
$$

where (a) is given by strong coordination, (b) is by the combination of the inequalities due to Pinsker and Breatgnolle-Huber, and (c) is Jensen's inequality. We now proceed to bound $\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(Q\|\hat{Q})\right]$ by observing that the average distortion between Q and $\hat{Q}$ is a linear function of the probabilities $\hat{Q}$, and the set $\hat{\mathcal{Q}} = \{\hat{Q} \in \Phi(\mathcal{H}) : d_{sem}(Q, \hat{Q}) \leq \epsilon\}$ satisfying the constraint is convex, and that by the definition of Distortion-Rate function, $\hat{Q}$ is the distribution in $\hat{\mathcal{Q}}$ minimizing $\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(\hat{Q}\|P)\right]$, where P is the pre-data coding distribution (see Theorem 16). Then, Theorem 11.6.1 in [69], also known as the Pythagorean Theorem for the Kullback-Leibler divergence, applies, and we can bound

$$
\begin{aligned}
\mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(Q\|\hat{Q})\right] &\leq \mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(Q\|P)\right] - \mathbb{E}_{C,S}\left[D_{\mathrm{KL}}(\hat{Q}\|P)\right] \\
&\overset{(a)}{=} R^* - R \\
&= \Delta_R,
\end{aligned}
$$

where (a) is given by Theorem 16. Combined together, the two inequalities provide Equation Eq. (2.39). Regarding Scheme 2, it is sufficient to notice that $R = I(S; \hat{H}^2) + I(S; \hat{S}^2|\hat{H}^2)$, from which Equation Eq. (2.40) follows.

We notice that this does not directly apply to d_{avg}, as the distribution Q^* solving Equation Eq. (2.34) and providing the rate is not, in general, the one used by Bob to sample actions. This is true for the scheme for d_{max}, in which common randomness is introduced.

3

Information in Distributed Decision Processes

In this second part of the thesis we are going to analyze the role of information in the different context of multi-agent and distributed decision processes. Specifically, in this learning framework we assume that there exists an underlying system that can be observed in discrete-time steps through a *context*, which can be used to infer the real *state* s of the system. In the single-agent formulation, based on the context (also referred to as state if there is no uncertainty in the observation), the agent interacts with the system taking a decision, or *action* a. Then, given the state and the taken action, the agent receives a signal, usually denoted as *reward*, which is sampled according to some underlying and unknown distribution. In the next step, the system moves to a new state s' according to a probability law which depends on the modeling assumption, and the process continues. Specifically, the new state s' can be sampled independently to s according to some identical distribution, i.e., each state is sampled in an independent and identically distributed (i.i.d.) fashion; or, there could be an unknown transition kernel $P(s'|s, a)$ expressing the probability of ending up in state s' when taking action a in state s. The goal for the agent is to find a policy π which is a possibly stochastic map indicating, in each state, which is the best action to choose in order to maximize the expected sum of the present plus future rewards. We'll refer to the first type of process as a contextual multi-armed bandit (CMAB) problem, in which actions are also referred to as arms, and we recognize that the second one is indeed a Markov decision process (MDP) [42]. We notice that such formulation can be utilized to model many real-world systems with a plethora of useful applications [129, 251, 135, 99].

The learning tasks considered in this chapter have the goal of producing an optimal policy π^*, which can indeed maximize the expected rewards for the agent, while interacting with the system. To this end the agent has to play with the system first to explore the effects its actions have on the underlying process (*exploration* phase), and then to optimize the decisions according to the acquired knowledge (*exploitation* phase). Now, depending on the underlying modeling assumption, many techniques have been proposed in the literature to find π^*. Specifically, when the problem is

formulated using the multi-armed bandit (MAB) framework, optimal algorithms have been well-studied and investigated [226]. In the case of MDP the problem is more complex as the actions taken by the agent influence not only the immediate reward, but also the states trajectories of the system, affecting indirectly the future rewards. Consequently, optimizing only for the best one-step reward may not be the optimal choice, as the action with highest expected reward in state s may lead to poor future states, and so to sub-optimal future performance. Reinforcement learning (RL) [233] is the tool that formulates the proper learning problem to solve the MDP, whose literature is vast and offers many solutions. Lately classical RL algorithms have been enhanced using Deep Learning (DL) leading to Deep Reinforcement Learning (DRL) [175], which is able to handle many complex problems by processing the state with deep neural networks (DNNs).

The problems analyzed in this chapter extend the formulation described in the previous paragraphs by considering scenarios in which the process of observing the system's state is physically separated from the action of interacting with the environment. Specifically, in Section 3.2 we defined the new *rate-constrained CMAB (RC-CMAB)* problem in which a *decision-maker* can observe the state of the system and has the computing capacity to optimize the policy π, while it has to communicate the actions to a set of agents through a rate-limited communication channel. The agents can then physically interact with the environment by taking the communicated actions affecting the underlying system. In this case the problem has been analytically solved and the theoretical limits reported. In Section 3.3 we then define the *remote Partially Observable Markov Decision Process (POMDP)*, which is the MDP counterpart of the *RC-CMAB* problem. In both situations the fundamental research question is to understand what is the relation between the observations and the optimal decisions. To this end, we will see that information-theoretic quantities will play a critical role in defining the limits of the problem, whose practical goal is to reduce as much as possible the information sent from the decision-maker to the agents in order to save network resources without damaging the final performance.

3.1 Related Work

We start by covering the literature which is mostly related to the problem. As the data exchanged in the communication network is strongly correlated with some training process, new data compression and representation methods should be considered to intelligently utilize network resources. The limits of lossy compression have been defined by rate-distortion theory [69], which provides the minimum number of bits needed to represent information given a target value for the maximum tolerable distortion between the original data, and the data reconstructed from the compressed version. However, in distributed machine learning (ML), the goal at the receiver is not to perfectly reconstruct the input data, but to perform some inference or training task based on it.

Communication in Multi-Agent Decision Processes. This chapter focuses on multi-agent distributed decision processes, where the goal is not to learn a function through supervised learning, but rather to optimize a policy, i.e., a map that, given an observation, provides a probability distribution over a set of feasible actions. However, the processes of collecting observations, train-

ing the policy and taking the actions are implemented by physically distributed entities [52], which require the design of compression and communication strategies to coordinate the training process. Similar configurations appear in the parallel training of a single logical agent, with the aim of accelerating the training process [176, 31, 65, 249, 226]. To this end, an important line of research is the study of the most valuable data to be shared among the agents in order to achieve cooperation, and thus convergence to better policies. Frameworks in [92, 231, 109, 149] admit cooperation in multi-agent reinforcement learning (MARL) among the agents through the transmission of signals, whose effects on the common task are differentiable with respect to the signal transmission decisions. In this case, the emergence of languages and cooperative messages is analyzed, with perfect communications links. The authors in [241] consider noisy communications, and propose a scheme to jointly learn good policies and optimal ways to code and transmit the actions over the communications channels. Other research efforts consider deep RL agents, and focus on compression schemes of neural networks (NNs), which are generally used to approximate value functions and target policies. For example, the authors in [276] adopt a knowledge distillation technique to reduce the complexity of a behavioral policy, while training a more complex target policy that serves as a teacher, in the context of parallel training [65]. In [167] and [30], the authors achieve model compression adopting pruning techniques [47, 123, 103] to reduce the size of an agent's NN-based policy. Both papers provide empirical studies of the performance of a single agent when trained with different pruning levels. In [167], an iterative process is used to prune the model, which is reduced after training the original (full size) NN. In our framework, training is performed over rate-limited communications channels, and there could be no opportunity to transmit the full information. The authors in [30] consider performing pruning exploiting an initial offline dataset, that is not present in our case. Moreover the pruning level, and so the needed rate to transmit the models, has to be fixed at the beginning of the training process. Our framework is similar to that of [104], where a server communicates actions to a pool of agents. However, in this case the analyzed link is the one between the agents and the server, and is used to send back the observed rewards. Moreover, no context (or state) is considered. In [17], the same MAB problem is solved cooperatively by a set of agents that can share some information about the best estimated actions. In this case communication is peer-to-peer with a deterministic rate, and again there is no contextual information. A related formulation to the problem in Section 3.2 has been proposed in [134], where the *Batched Thompson Sampling* algorithm is introduced. In this case the goal is to reduce the number of policy updates, thus reducing the computational complexity of the algorithm. Rounds, i.e., single time steps, are grouped into batches, and within one batch arms are pulled without updating the sampling policy. Similarly, in the RC-CMAB formulation one round can be considered as a batch, given that the N agents operate in parallel. However, in [134], the *batch size*, i.e., the number of samples without policy updates, can be optimized by the algorithm and is not fixed during the training, i.e., different batches may have different sizes. On the contrary, in our case the batch size is fixed to N, and is given by the environment. Moreover, in [134] the authors do not consider the contextual case, and there are no communications constraints when pulling arms.

Communications in Remote Control. The specific requirements of distributed and remotely controlled systems have focused the research community's attention towards communication systems which must provide updated information to enable real-time high-level tasks such as inference, tracking or control. Although metrics such as Age of Information (AoI) [272] represent a major improvement with respect to latency and packet loss, they are still limited, as they assume that the quality of the information available at the receiver degrades deterministically with time, most commonly (but not necessarily [139]) in a linear fashion. However, more sophisticated systems can also take into consideration the current state of the system in order to decide whether and when to update the status of the receiver. Metrics such as Urgency of Information (UoI) [278] and Value of Information (VoI) [260] incorporate state information in their definition and are thus aware of the intrinsic value of potential updates. Other context-aware indices to measure the non-linear time-varying importance and the non-uniform context-dependence of the status information have also been proposed [278]. The authors of [93] considered a system in which a transmitter monitors the status of a system and updates the controller, providing status information. Then a constrained MDP is formulated to minimize the cost of actuation and simultaneously guarantee a target communication rate. Both works show significant improvements with respect to other metrics such as AoI.

The Levels of Communications. The main goal of classical communication theory is to build reliable systems for the accurate and efficient transmission of data. However, in the preface to Shannon's seminal work [222], Warren Weaver already envisioned two more complex levels of communication beyond the simple transmission of bits. Classical communications are then included in Level A, or the *technical problem*, which concerns itself with the accurate and efficient transmission of arbitrary raw data. Level B, or the *semantic problem*, is to find the most effective way to convey the meaning of the message, even when irrelevant details are lost or misunderstood, while Level C, also called the *effectiveness problem*, deals with the resulting behavior of the receiver: as long as the receiver takes the optimal decision, the effectiveness problem is solved, regardless of the quality of the received information. While the Level B and C problems attracted limited attention for decades, the explosion of Industrial Internet of Things (IIoT) systems has drawn the research and industrial communities toward semantic and effective communication [205], optimizing remote control processes under severe communication constraints beyond Shannon's limits on Level A performance [100]. In particular, the effectiveness problem is highly relevant to robotic applications, in which independent mobile robots, such as drones or rovers, must operate based on information from remote sensors. In this case the sensors and the cameras act as the transmitter in a communication problem, while the robot is the receiver: by solving the Level C problem, the sensors can transmit the information that best directs the robot's actions toward the optimal policy [229]. We can also consider a case in which the robot is the transmitter, while the receiver is a remote controller, which must get the most relevant information to decide the control policy [255]. At the same time, the development of learning-based coding schemes has allowed communication system designers to move beyond packet error as the key coding performance metric, exploiting semantic considerations. Joint source-channel coding for wireless image transmission is implemented in [50,

265], and the encoder-decoder pair is parameterized by a NN, whose architecture may vary. This approach can be used to maintain the semantic information contained in the transmitted data, while improving compression performance. Semantic information at the receiver can be used to solve different tasks. Effective communication [241] can be seen as an extension of this, in which the task involves the receiver taking actions and possibly altering the information that the transmitter is communicating. Effective communication differs from semantic communication mostly because the "semantic" content which has to be preserved in the communicated messages is not explicit. Moreover, control tasks have a temporal component that must be taken into account, as investigated in [241]. The scenario considered in Section 3.3 is a two-agent POMDP in which one agent communicates and the other agent interacts with the environment, using DRL to solve the joint problem and encoding the information. A distributed perception scenario, in which multiple sensors communicate to a single robot, is considered in [170] and solved using MARL, showing that joint training improves the performance of the system, particularly when communication is severely constrained. While past works aimed at specific scenarios and objectives, the solution proposed in Section 3.3 proposes a novel DRL approach that combines status updates with an adaptive coding scheme and can be easily adapted to operate on any levels of communication (A, B and C).

Connections with Psychology. Interestingly, the authors of [146] study similar frameworks to justify the concept of information bottleneck in RL [86, 121] from a psychological perspective. In this work, the rate-constrained channel is compared to resource-limited policy storing systems, e.g., the brain and noisy storage devices, commenting on the trade-off between policy complexity, i.e., level of correlation between states and actions, and policy performance, i.e., obtained reward. In these studies the single-agent case is considered, and the formulation is somewhat generic with qualitative and empirical considerations, without properly framing the underlying learning problem. In this older study [67], the authors performed a set of behavioral experiments that resemble the CMAB problem, recording correlations between performance variance, i.e., action stochasticity, and model capacity, i.e., work memory, in humans.

3.2 The Rate-Constrained Remote Contextual Multi-Armed Bandit Problem

With this work, the aim is to study this relation within the proper information-theoretic setting, relating it to the learning task of CMAB, and highlighting the theoretical trade-off between channel rate and learning performance. Moreover, we study practical ways to compress policies having limited impact in the training process.

3.2.1 Introduction

The past decade has seen a transition from centralized computing solutions, based on the cloud, to more distributed systems, also known as multi-access edge computing [216], mainly in response

to the emerging paradigm of Internet of everything, where data is generated, processed and consumed by a network of connected nodes with diverse storage and computing capabilities. Another paradigm shift involves the adoption of ML as a core technology for future services. ML techniques, mainly driven by deep NNs, achieve state-of-the-art performance in many practical applications such as computer vision, speech recognition, and automatic control [60].

The focus of this work is to study the information-theoretic limits of distributed learning in the specific context of multi-agent contextual multi-armed bandits (CMABs). CMABs model decision-making problems in which an agent interacts with an environment in sequential rounds. At each round, the agent observes a context, which contains side information on the environment, and has to pull one out of K arms. Based on the observed context and pulled arm, the environment returns a reward, which is sampled according to an unknown distribution. The goal of the agent is to optimize an arm selection strategy to maximize the average sum of obtained rewards. In this scenario, the contexts of N agents are available to a central *decision-maker* that has to inform a remote entity, called the *controller*, on the arms the agents should pull. Depending on the scenario, the controller either directly pulls the arms, or provides the indices of the suggested arms to the agents, that can physically interact with the environment. However, we assume that there is a rate-limited communication channel between the decision-maker and the controller, which limits the number of bits the decision-maker can use to convey the intended arms at each round. This framework can model standard recommendation systems, where a content owner observes some client-dependent features on one side, and must communicate the recommendations to a separate entity, which in turn proposes the recommended items to the clients. Another related engineering problem is that of distributed training of RL policies. Particularly in deep RL, complex policies are usually trained exploiting parallel simulations involving many agents, which can simultaneously collect data to speed up the policy training convergence, running at the central server. Consequently, the communication link between the *decision-maker* and the *controller*, over which potentially high-dimensional data, i.e., the policy/observation/actions, must be conveyed, may represent the bottleneck of the system. Further analysis may also consider limited communication budget in the feedback link, over which the rewards are transmitted. The purpose of this work is to formulate the novel RC-CMAB framework, and to study the theoretical limits, i.e., in the regime of infinite agents, of this communication problem when the available rate is below what would be required to perfectly convey the decision-maker's policy at each iteration. Then, we provide practical lossy policy compression schemes, that take into account the learning objective when specifying the distortion metric to be used, and can be applied in the more practical scenario with a finite number of agents. We then draw connections between our framework and the *information bottleneck* [238, 98] and *maximum entropy reinforcement learning* [152, 101] approaches. Finally, we report numerical results in support of our analyses.

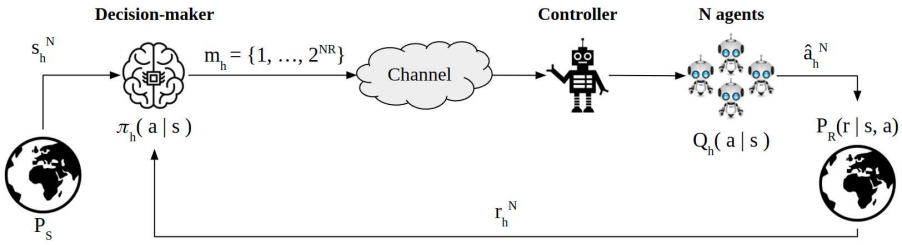


Figure 3.1: The RC-CMAB problem formulation.

3.2.2 Problem Formulation

The Contextual Multi-Armed Bandit (CMAB) Problem

The standard single-agent CMAB problem considers an agent interacting with the environment by pulling arms upon the observation of some contextual information, and receiving a reward based on the observed context and pulled arm. Specifically, at each round $t = 1, \ldots, T$, the environment samples a context $s_t \in \mathcal{S}$ following distribution P_S, where $\mathcal{S}$ is a finite set containing all possible contexts. We assume that all contexts are observable, i.e., $\forall s \in \mathcal{S},\ P_S(s) > 0$. When observing s_t, the agent chooses an arm $a_t \in \mathcal{A} = \{1, \ldots, K\}$, with probability $\pi_t(a_t|s_t)$. Given the pair (s_t, a_t), the environment returns a stochastic reward $R(s_t, a_t)$ sampled according to $P_R(r|s_t, a_r)$, which is an unknown and stationary distribution that characterizes the reward statistics, and depends on the sampled context and the arm pulled by the agent. We then define $\mu(s, a) = \mathbb{E}_{P_R}[R(s, a)]$, and further assume that $R(s, a) \in [0, 1]\,,\ \forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$. Moreover, we assume that the reward distributions belong to the exponential family[1], as also detailed in [202], Assumption 1. The policy $\pi_t(a_t|s_t)$ employed by the agent is a map $\pi_t : \mathcal{H}^{t-1} \times \mathcal{S} \to \Delta_K$, where Δ_K denotes the $(K-1)$-simplex, containing all possible distributions over the set of K arms, and $\mathcal{H}^{t-1}$ is the history, containing all possible observations, arms and rewards collected until round $t-1$, i.e., the element $H(t-1) \in \mathcal{H}^{t-1}$ is $H(t-1) = \{s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}\}$. The goal for the agent is to optimize the policy π_t in order to maximize the average sum of received rewards or, equivalently, to minimize the Bayesian regret

$$\mathrm{BR}(\pi, T) = \mathbb{E}\left[\sum_{t=1}^{T} \mu(s_t, a^*(s_t)) - \mu(s_t, A_t)\right], \tag{3.1}$$

where A_t is the arm pulled by the agent in round t sampled according to $\pi_t(a|s_t)$, and $a^*(s_t) = \mathrm{argmax}_{a \in \mathcal{A}}\, \mu(s_t, a)$ is the optimal arm in round t, i.e., the one that maximizes the average reward in context s_t. If $a^*(s)$ is not unique, it represents an arbitrarily chosen arm among the optimal ones. Here the expectation is taken with respect to the state, action and problem instance distributions.

[1]If $\boldsymbol{\theta} = (s, a)$, we can write $P_R(r|\boldsymbol{\theta}) = b(r)\exp(\eta(\boldsymbol{\theta})T(r) - A(\boldsymbol{\theta}))$, where b, T, and A are known functions, and $A(\boldsymbol{\theta})$ is assumed to be twice differentiable.

Rate-Constrained CMAB

We consider a system in which N agents have to solve the same realization of a CMAB problem, where each agent observes an independent context, distributed according to P_S, and depicted in Fig. 3.1. The agents can only interact with the environment pulling arms, whereas the contexts are observed by a remote decision-maker. The decision-maker communicates with the agents through a controller, that is in charge of receiving instructions from the decision-maker, and informing the agents accordingly. However, the decision-maker can transmit information to the controller through a rate-constrained channel, which imposes a constraint on the number of bits per agent the decision-maker can transit to the controller. Consequently, at each system round, the environment samples N contexts $\{s_{j,n}\}_{n=1}^{N}$, i.e., one per agent, which are observed by the decision-maker, which, in turn, needs to decide and communicate to the controller the N arms $\{a_{j,n}\}_{n=1}^{N}$ to be pulled by the agents.

Then, the decision-maker exploits the knowledge accumulated until system round j, and encoded in the variable $H(j-1) = \left\{\{s_{1,n}, a_{1,n}, r_{1,n}\}_{n=1}^{N}, \ldots, \{s_{j-1,n}, a_{j-1,n}, r_{j-1,n}\}_{n=1}^{N}\right\} \in \mathcal{H}^{(j-1)}$ to optimize its policy π_j. However, in our setting, the decision-maker can interact with the controller only through a rate-constrained communication channel, which may not allow to transmit all the intended arms to the agents. Consequently, the problem is to communicate the arm distribution, i.e., the policy $\pi_j(a|s)$, which depends on the specific context realizations observed in round j, to the controller within the available communication resources while inducing the minimal impact on the performance of the learning algorithm. To this end, the decision-maker employs a function $f_j^{(N)} : \mathcal{H}^{(j-1)} \times \mathcal{S}^N \to \{1, 2, \ldots, B\}$ mapping the knowledge acquired up to round $j-1$, together with the agents' contexts, to a message index to be transmitted over the channel. At the receiver, the controller adopts a function $g_j^{(N)} : \{1, 2, \ldots, B\} \to \mathcal{A}^N$ to decode from the received message the N arms to be pulled by the agents. In general, both functions $f_t^{(N)}$ and $g_t^{(N)}$ can be stochastic. We then define the Bayesian system regret as

$$\mathrm{BR}\left(J, \left\{f_j^{(N)}, g_j^{(N)}\right\}\right) = \mathbb{E}\left[\sum_{j=1}^{J}\sum_{n\in\mathcal{N}} r(s_{j,n}, a^*(s_{j,n}) - r(s_{j,n}, g_{j,n}(m_j))\right], \tag{3.2}$$

where $g_{j,n}(m_j)$ is the arm pulled by agent n during round j decoded from the message $m_j = f_j^{(N)}\left(H(j-1), s_j^N\right)$, and $s_j^N \in \mathcal{S}^N$ is the vector containing the contextual information in round j for all the N agents. The goal is to specify the encoding and decoding functions, $f_j^{(N)}$ and $g_j^{(N)}$, to minimize the Bayesian system regret in Eq. (3.2). More specifically, the goal is to obtain a regret which is sub-linear in J, possibly achieving the same performances obtained by standard CMAB solutions [148]. For a problem with N agents, a rate R is said to be achievable if there exist functions $\left\{f_j^{(N)}, g_j^{(N)}\right\}_{j=1}^{J}$ with rate $\frac{1}{N}\log_2 B \leq R$, and Bayesian system regret

$$\lim_{J\to\infty} \frac{\mathrm{BR}\left(J, \left\{f_j^{(N)}, g_j^{(N)}\right\}\right)}{J} = 0. \tag{3.3}$$

Note that when there is no communication channel from the decision-maker to the controller,

i.e., $B = 0$, sub-linear regret is not possible, since the agents cannot learn on their own without observing the contexts and rewards. On the other hand, when B is sufficiently large, i.e., $B \geq N \log_2 K$, any desired arm sequence can be conveyed to the controller, and the problem becomes a distributed CMAB problem with N parallel agents. Our goal is to identify the minimal communication needed from the decision-maker to the controller that makes sub-linear regret feasible. We would like to emphasize that the introduced RC-CMAB problem differs from the standard CMAB formulation in two aspects: First, at each round j, the decision-maker pulls N parallel arms through the agents exploiting π_j, which is updated at the end of each round. This is similar to the batch MAB formulation presented in [134], in which the policy can be updated only every N arm pulls. Second, given the available rate R, the decision-maker may not be able to convey the exact sequence of arms $\{a_{j,n}\}_{n=1}^{N}$ sampled from π_j, and instead must send a compressed version, which may result in some agents to pull sub-optimal arms. We highlight that this is a lossy compression problem; however, unlike classical lossy source coding problems, the goal is not to send a sequence of arms with highest average fidelity, but to enable the agents to pull the arms that would result in a sub-linear regret.

3.2.3 Theoretical Limits

In this section, we provide a theoretical analysis of the regret bound achievable by the Thompson Sampling (TS) strategy, and the minimum rate required to achieve sub-linear regret.

TS Performance

We first provide the regret performance of the TS algorithm, when there is no constraint on the available rate to transmit the intended arms. In this work, we analyze the general case in which no prior structure is assumed between the optimal policies and the different contexts, and so we consider the simplest implementation of one MAB agent for each context $s \in \mathcal{S}$. The TS algorithm adopts a Bayesian strategy, estimating the distribution $p^{s,a}(\mu)$ of the reward mean $\mu(s,a) \in [0,1]$ in each round j with $p_j^{s,a}(\mu)$. When observing the context s_j, it samples $\hat{\mu}_j(s_j,a) \sim p_j^{s,a}(\mu)$, $\forall a \in \mathcal{A}$, and pulls the arm $a_j = \text{argmax}_{a\in\mathcal{A}}\{\hat{\mu}_j(s_j,a)\}$. In RC-CMAB the decision-maker adopts the described sampling strategy for each agent $n \in \{1,\dots,N\}$. After receiving all the rewards $\{r_{t,n}\}_{n=1}^{N}$, the decision-maker updates its belief on $\mu(s,a)$ optimizing the posteriors $p_j^{s,a}(\mu)$. The variance of the posterior distributions is exploited to perform exploration. This algorithm is well studied, and is known under the name of Thompson Sampling (TS) [237]. The TS algorithm implicitly induces a probability distribution $\pi_j(a|s)$ over the arms that can be computed as

$$\pi_j(a|s) = \int_{\mathbb{R}} p_j^{s,a}(\mu) \prod_{k=1,k\neq a}^{K} P_j^{s,k}(\mu) d\mu,$$

where $P_j^{s,k}(\mu)$ is the cumulative distribution function (CDF) of $\mu(s,k)$, and the random variables $\mu(s,a)$ are independently distributed. We will call $\pi_j(a|s)$ the target policy, i.e., the one that the decision-maker would like to convey to the controller. However, in RC-CMAB the constraint on

the rate of the communication channel may not allow to sample the arms according to $\pi_j(a|s)$, as explained in Sec. 3.2.2. In this case, the problem is that the decision-maker updates the posteriors $p_j^{s,a}$, and so $\pi_j(a|s)$, using the TS algorithm, but can only sample with an approximate policy $Q_j(a|s)$, whenever the available rate is not sufficient to convey $\pi_j(a|s)$.

Regret Bounds

We now report the performance of the TS algorithm, when the available rate is sufficient to perfectly convey the arms from the decision-maker to the controller in each round $j = 1, \dots, J$. To this end, we align the parallel interactions between the agents and the environment in time, and consider virtual rounds $t \in \{1, \dots, J \cdot N\} = \mathcal{T}$, as if a single agent, i.e., the decision-maker, were playing a sequential CMAB game, with the constraint that the policy can be updated only every N steps, i.e., at the end of each round. We observe that the order used to align the agent interactions does not affect the analysis, as the agents all play the same policy, and interact with i.i.d. replicas of the same environment, leading the two formulations to be mathematically equivalent. Consequently, by providing the results as a function of the virtual rounds t, we consider the total number of interactions the agents have with the environment, which is consistent with the usual MAB notation.

Theorem 20 (TS Bayesian System Regret)**.** *The finite-time Bayesian system regret of TS is upper bounded by*

$$BR(\pi^{TS}, T) \leq 2SKN + 4\sqrt{(2 + 6\log T)\, SKNT}, \tag{3.4}$$

and the asymptotic regret is

$$BR(\pi^{TS}, T) \in \mathcal{O}\left(\sqrt{KST \log T}\right). \tag{3.5}$$

Proof. See Appendix 3.4.1. □

We observe that, in the finite-time analysis, an additional term $\sqrt{N}$ appears, with respect to the standard single-agent performance [148]. This is a consequence of the fact that, in one round j, N arms are pulled in parallel, without updating the policy. This effect has highest impact during the first rounds, as the policy has not converged yet, and so sub-optimal arms are pulled in parallel. In the long term, the effect vanishes. This result is consistent with the analysis in [134], with the difference that what the authors called batch, in our scenario is the parallel execution of the N agents, and so in our case the batch size is fixed to N and can not be optimized. Consequently, in the finite-time upper bound we obtain a factor $\sqrt{N}$, that replaces the factor $\sqrt{\alpha}$ obtained in [134], where α is the so-called *batch growing factor* [134]. The factor S is introduced as we consider the CMAB problem.

Rate-Distortion Function for Communicating Policies

We now present the minimum rate needed to transmit a policy, i.e., the arms $a_j^N = (a_{j,1}, \dots, a_{j,N})$ to be sampled for each agent according to $\pi_j(a|s)$, and conditioned on the observed context vec-

tor $s_j^N = (s_{j,1}, \dots, s_{j,N})$, when a specific distortion function is adopted to measure the discrepancy between the seeking sequence $z_j^N = ((s_{j,1}, a_{j,1}), \dots, (s_{j,N}, a_{j,N}))$, and the sequence $\hat{z}_j^N = ((s_{j,1}, \hat{a}_{j,1}), \dots, (s_{j,N}, \hat{a}_{j,N}))$, where $\hat{a}$ indicates the arms decoded by the controller based on the received message m_j, as indicated in Sec. 3.2.2. In short, $\hat{a}_j^N$ is the vector containing the arms actually pulled by the agents. Given the underlying learning problem, the quality metric for the vector $\hat{a}_j^N$ should not be based on a per-symbol distance, but rather on the sampling probability distributions. Indeed, to obtain sub-linear regret, what interests us is the probability of sampling specific sequences. Consequently, the distortion function $d(\hat{Q}_{\hat{z}^N}, P_{SA})$ compares the empirical distribution $\hat{Q}_{\hat{z}^N}$ of $\hat{z}^N$ with P_{SA}, which is the joint distribution $P_{SA} = P_S(S) \cdot \pi(A|S)$. In the sequel, we omit to explicitly write the round index j, as the analysis does not depend on it. We consider the particular case in which the distortion measure $d(\hat{Q}_{\hat{z}^N}, P_{SA})$ respects the following properties: it is 1) nonnegative; 2) upper bounded by a constant D_{max}; 3) continuous in P_{SA} at $\hat{Q}_{\hat{z}^N}$; 4) convex in P_{SA}, and such that 5) $d(\hat{Q}_{\hat{z}^N}, P_{SA}) = 0 \iff \hat{Q}_{\hat{z}^N} = P_{SA}$. Given the assumptions above, the authors of [140] provide the rate-distortion function $R(D)$, i.e., the minimum rate $R = \frac{\log_2 B}{N}$ bits per symbol such that $\mathbb{E}_{Q_{SA}}[d(\hat{Q}_{\hat{z}^N}, P_{SA})] \leq D$, in the limit when N is arbitrarily large. Here the expectation is taken with respect to the distribution $Q_{SA} = P_S(S)Q(A|S)$, where $Q(A|S)$ is the sampling policy decoded by the controller from the message sent by the decision-maker. The solution is given by

$$R(D) = \min_{Q_{A|S}: d(Q_{SA}, P_{SA}) \leq D} I(S; A). \tag{3.6}$$

As we can see, in the asymptotic limit when $N \to \infty$, the problem admits a single-letter solution, which also serves as a lower bound for the finite agent scenario. Let R_{π_j} denote the rate required to perfectly convey the policy π_j , i.e., with zero distortion. From Eq. (3.6), we can see that $R_{\pi_j} = I(S; A)$, where the mutual information is computed under P_{SA} dictated by the policy.

Achievable Rate

We now state the rate condition under which it is possible to achieve sub-linear regret. In the analysis, the available rate R is considered fixed in each round j. First of all, we denote with $H(A^*)$ the entropy of the arms under the marginal distribution $\pi^*(a) = \sum_s P_S(s)\pi^*(a|s)$, where π^* is the optimal policy, i.e., the one that selects, $\forall s \in \mathcal{S}$, the arm $a^* = \text{argmax}_{a \in \mathcal{A}} \mu(s, a)$. We start by stating the following Lemma, which provides a rate limit below which it is not possible to achieve sub-linear regret.

Lemma 21. *If $R < H(A^*)$, it is not possible to achieve sub-linear Bayesian system regret.*

Proof. See Appendix 3.4.1. □

The following Lemma provides the achievability part.

Lemma 22. *If $R > H(A^*)$, then it is possible to achieve sub-linear Bayesian system regret in the limit $N \to \infty$.*

Proof. See Appendix 3.4.1. □

We can see that, due to Lemma 22, even if for some round j, $R_{\pi_j} > R$, as long as $R > H(A^*)$, it is still possible to achieve sub-linear regret. According to the definition in Eq. (3.3), this implies that, as $N \to \infty$, any rate $R > H(A^*)$ is achievable, while any rate $R < H(A^*)$ is not achievable. The entropy of the marginal $\pi^*(a)$ is thus the fundamental information-theoretic limit of the problem to achieve sub-linear regret.

3.2.4 Policy Compression

We are now ready to study compression strategies to deal with the case in which it is not always possible to convey the policy with zero distortion, i.e., $\exists\ j$ s.t. $R_{\pi_j} > R$. In such cases, it is not clear which is the message m_j the decision-maker should transmit to the controller. As a consequence of Eq. (3.6), when $R_{\pi_j} > R$, the sampling policy Q_j adopted by the controller may differ from π_j. In [202], the authors provide some theoretical guidelines to construct approximate sampling policies to make the posteriors, i.e., $\pi(a|s)$, converge to the optimal one achieving sub-linear regret, even when an agent is sampling with a different policy $Q(a|s)$. In particular, they studied the case in which the sampling distribution Q differs from the target posterior π, using the α-divergence $D_\alpha(\pi, Q)$ as distortion measure, which is defined as

$$D_\alpha(\pi, Q) = \frac{1 - \int \pi(x)^\alpha Q(x)^{1-\alpha} dx}{\alpha(1-\alpha)}. \tag{3.7}$$

We now provide two theoretical compression schemes that adopt the forward KL divergence $D_{\alpha\to 0}(\pi, Q) = D_{KL}(\pi||Q)$, and the reverse KL divergence $D_{\alpha\to 1}(\pi, Q) = D_{KL}(Q||\pi)$, as distortion functions, which are the two cases considered also in [202]. We remember that, for two discrete distributions p and q such that p is absolutely continuous with respect to q, i.e., if $x \in \mathcal{X}$ is such that $q(x) = 0$ implies $p(x) = 0$, the KL divergence is defined as $D_{KL}(p||q) = \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$.

Another reason to adopt these two metrics is related to the fact that it is possible to bound, in each round j, the gap between the expected reward of the target policy π_j, and that obtained by using the approximate policy Q_j. To this end, we denote by $\mu^\pi(s, a)$ the average reward obtained in context s when using policy π, i.e., $\mu^\pi(s, a) = \mathbb{E}_{\pi(a|s)}[\mu(s, a)]$, and find

$$\mathbb{E}_{P_S}\left[\left|\mu^{\pi_j}(S, A) - \mu^{Q_j}(S, A)\right|\right] = \sum_{s\in\mathcal{S}} P_S(s) \sum_{a\in\mathcal{A}} \mu(s, a)|\pi_j(a|s) - Q_j(a|s)| \tag{3.8}$$

$$\overset{(a)}{\leq} \sum_{s\in\mathcal{S}} P_S(s) \sum_{a\in\mathcal{A}} |\pi_j(a|s) - Q_j(a|s)| \tag{3.9}$$

$$= \sum_{s\in\mathcal{S}} P_S(s)||\pi_j(\cdot|s) - Q_j(\cdot|s)||_1 \tag{3.10}$$

$$\overset{(b)}{\leq} C \cdot \mathbb{E}_{P_S}\left[\sqrt{D_{KL}\left(\pi_j(\cdot|S)||Q_j(\cdot|S)\right)}\right], \tag{3.11}$$

where (a) holds because $\mu(s, a) \in [0, 1]$ by assumption, (b) is the Pinsker's inequality, and C is

a constant that depends on the base of the logarithm in the divergence, e.g., $C = \sqrt{\frac{1}{2 \ln 2}}$ if in base 2. We notice that, by swapping the roles of the two distributions in the last inequality, it is possible to obtain the version with the reverse KL divergence.

Observation. It is known that by minimizing the forward KL divergence $D_{KL}(\pi||Q)$, we obtain a more "spread" solution Q, that tends to cover the whole domain of π. Indeed, $\forall x \in \mathcal{X}$ s.t. $\pi(x) > 0$, a penalty is added whenever the two distributions differ, i.e., the function $\log \frac{\pi(x)}{Q(x)}$ is weighed by $\pi(x)$. On the other hand, in the reverse KL divergence, the log function is weighed by distribution Q. This means that, if $Q(x) = 0$, there is no incurred penalty for not approximating the policy $\pi(x)$ in x, leading to a solution that puts mass around the peaks of π. Consequently, the exploration-exploitation trade-off is usually biased towards a more exploration-seeking solution when minimizing $D_{KL}(\pi||Q)$, and to a more exploitation-seeking solution when minimizing $D_{KL}(Q||\pi)$.

Reverse KL Divergence

In this case, the adopted distortion function is $d(Q_{SA}, \pi_{SA}) = \mathbb{E}_{P_S}\left[D_{KL}(Q(\cdot|S)||\pi(\cdot|S))\right]$. In the following lemma, we provide the shape of the policy $Q(a|s)$ that can achieve the minimum in Eq. (3.6), i.e., the policy that minimizes the required rate while meeting the constraint on the reverse KL divergence.

Lemma 23. *Given the constraint on the reverse KL divergence $D_{KL}\left(Q||\pi\right) \leq \delta$, the policy that achieves the minimum in Eq. (3.6) is*

$$Q_\lambda(a|s) = \frac{\tilde{Q}(a)^\lambda \pi(a|s)^{1-\lambda}}{Z}, \tag{3.12}$$

where $\lambda \in [0, 1]$ is such that $D_{KL}\left(Q_\lambda||\pi\right) = \delta$, $\tilde{Q}(a)$ is the marginal, i.e., $\tilde{Q}(a) = \sum_{s \in \mathcal{S}} P_S(s) Q_\lambda(a|s)$, and Z is the normalization factor.

Proof. See Appendix 3.4.1. □

We can see that, when $\lambda = 0$, we have $Q(a|s) = \pi(a|s)$, and so $R_Q = R_\pi = I(S; A)$, where the mutual information is computed with respect to P_{SA}. As λ increases from 0 to 1, the policy Q tends to be more similar to the marginal $\pi(a) = \sum_{s \in \mathcal{S}} P_S(s)\pi(a|s)$, which requires zero rate. Indeed, when $\lambda = 1$, the marginal $\pi(a)$ does not require context information, and so the agents would sample arms regardless of the contexts $s_{j,n}$.

Forward KL Divergence

We consider $d(Q_{SA}, \pi_{SA}) = \mathbb{E}_{P_S}\left[D_{KL}(\pi(\cdot|S)||Q(\cdot|S))\right]$. As for the reverse case, we report the shape of the optimal compressed policy.

Lemma 24. *Given the constraint on the forward KL divergence $D_{KL}(\pi||Q) \leq \delta$, the policy that achieves the minimum in Eq. (3.6) is*

$$Q_\lambda(a|s) = \frac{\lambda\pi(a|s)\ \mathbf{W}_0\left(\frac{\lambda\pi(a|s)}{\tilde{Q}}\right)}{Z} \tag{3.13}$$

where λ is such that the maximum distortion is achieved with equality $D_{KL}(\pi||Q_\lambda) = \delta$, $\mathbf{W}_0(\cdot)$ is the Lambert function [147], $\tilde{Q}(a)$ is the marginal, i.e., $\tilde{Q}(a) = \sum_{s\in\mathcal{S}} P_S(s)Q_\lambda(a|s)$, and Z the normalization factor.

Proof. See Appendix 3.4.1. □

We observe that the above compressed policies can achieve the minimum in Eq. (3.6) as $N \to \infty$, characterizing the information-theoretic limit and serving as a lower bound for more practical schemes that can work for finite N.

Remark. We notice that, in general, $D_{KL}(p||q)$ always satisfies conditions 1) and 3) − 5) defined in Section (3.2.3), but not condition 2), i.e., it may be unbounded. We now define $\sigma_q := \min_{x\in\mathcal{X}:q(x)>0} q(x)$. If p is absolutely continuous with respect to q, by the *inverse Pinsker's inequality* we have

$$D_{KL}(p||q) \leq \frac{2\mathrm{TV}(p,q)^2}{\sigma_q} \leq \frac{8}{\sigma_q} = D_{max},$$

where $\mathrm{TV}(p,q)$ indicates the total variation distance between the distributions p and q. To meet this requirement, we further assume $\exists\epsilon > 0 : \pi(a|s) \geq \epsilon$, $\forall s \in \mathcal{S}$ and $a \in \mathcal{A}$, whenever the compression scheme has to be applied. This additional assumption is reasonable in our context, as the target policy π_j needs exploration during training. Once π_j has converged to the optimal policy (see Appendix 3.4.1), further compression would lead to sub-linear regret.

Practical Coding Scheme

To find practical coding schemes, we propose a solution that is based on the idea of context reduction, and computes compact context representations. In essence, the decision-maker constructs a message containing the new context representations $\hat{s}(s) \in \hat{\mathcal{S}}$ of s, one for each agent, and sends it over the channel. Once the agents have received the message, they can sample the arms according to a common policy $Q_{\hat{s}}(a|\hat{s})$, which is defined on the compressed context space $\hat{\mathcal{S}}$. If the rate constraint imposes B bits per agent, it means that it is possible to transmit at most 2^B different contexts to each agent. The idea is to group the contexts into $2^B = M$ clusters $\hat{s}_1, \dots, \hat{s}_M$, minimizing $d(Q_{\hat{S}A}, P_{SA})$, where $Q_{\hat{S}A}$ is the new policy defined on the compressed contexts $\hat{s}(s) \in \hat{\mathcal{S}}$. Again, we avoid to explicitly write the round index j, as the scheme does not depend on it.

To find the clusters and relative policies we employ the well-known Lloyd algorithm [168], which is an iterative process to group states into clusters. First of all, knowing the policy π, the decision-maker maps each state $s \in \mathcal{S}$ into a K-dimensional point $\boldsymbol{\alpha}^{\boldsymbol{p}} = \pi(\cdot|s) \in \Delta_K$, finding

$|\mathcal{S}| = S$ different points $\boldsymbol{\alpha^1}, \ldots, \boldsymbol{\alpha^S}$. Then, it generates $2^B = M$ random points $\boldsymbol{\mu^1}, \ldots, \boldsymbol{\mu^M} \in \Delta_K$ as initial centroids, i.e., representative policies, and iterates over the following two steps:

1. Assign to each point $\boldsymbol{\alpha^p}$ the class $c^* \in \{1, \ldots, M\}$ such that $c^* = \arg\min_c \mathbb{E}_{P_S}\left[D_\alpha(\boldsymbol{\mu^c}, \boldsymbol{\alpha^p})\right]$, i.e., minimizing the average α-divergence between the representative $\boldsymbol{\mu^{c^*}}$ and the original policy, which is the point $\boldsymbol{\alpha^p}$. For each cluster c, we now denote by $\mathcal{S}_c$ the set containing the contexts associated to the policies in that cluster.

2. Update $\boldsymbol{\mu^1}, \ldots, \boldsymbol{\mu^M}$ such that $\boldsymbol{\mu^c} = \text{argmin}_{\boldsymbol{\mu} \in \Delta_K} \sum_{s \in \mathcal{S}_c} P(s) D_\alpha(\boldsymbol{\mu}, \pi(\cdot|s))$, which is still a convex optimization problem when using $\alpha = 0$ or $\alpha = 1$, and can be solved applying the Lagrangian multipliers. The solution is

$$\boldsymbol{\mu^c} = \frac{\prod_{s \in \mathcal{S}_c} \pi(\cdot|s)^{\frac{P(s)}{A(\mathcal{S}_c)}}}{Z}, \tag{3.14}$$

when using $D_{\alpha \to 0}(\boldsymbol{\mu^c}, \pi(\cdot|s)) = D_{KL}(\boldsymbol{\mu^c} || \pi(\cdot|s))$, and

$$\boldsymbol{\mu^c} = \frac{\sum_{s \in \mathcal{S}_c} P(s) \pi(\cdot|s)}{Z}, \tag{3.15}$$

when using $D_{\alpha \to 1}(\boldsymbol{\mu^c}, \pi(\cdot|s)) = D_{KL}(\pi(\cdot|s) || \boldsymbol{\mu^c})$. Here, the product is to be considered element-wise, $A(\mathcal{S}_c)$ is the sum of the contexts probabilities in $\mathcal{S}_c$, i.e., $A(\mathcal{S}_c) = \sum_{s \in \mathcal{S}_c} P(s)$, and Z is the normalizing factor. After computing the new centroids, we go back to step 1). The derivations of Eq. (3.14) and Eq. (3.15) are provided in Appendix 3.4.1 and Appendix 3.4.1, respectively.

The process continues until the new solution does not decrease the average distortion between the cluster policies and the target ones.

Observation. Note that the controller is assumed to know the M policies from which it samples the arms of the agents. This can be transmitted at the beginning of each round. In this case, the scheme is efficient as long as $N \log_2 K \gg BP \log_2 K$, where P is the number of bits used to represent the values of the Probability Mass Function (PMF) $Q_{\hat{s}}(\cdot|\hat{s})$. For this reason, we provide a scheme where the new policy is updated not at every transmission, but only when the new target π has changed considerably. In particular, if we denote with π^{cls} the policy defined over the compressed context representation, with π^{last} the last policy used to compute π^{cls}, and with π the updated target policy, we compute and transmit π^{cls} every time $D_\alpha(\pi^{last}, \pi)$ exceeds a threshold ζ.

Algorithm Complexity. The complexity of the algorithm is related to solving the k-means problem, which is known to be NP-hard. Consequently, the solution relies on state-of-the-art heuristic clustering schemes, which have complexity $\mathcal{O}(SKMI)$ [228], where S is the number of states, K is the number of total actions, M is the number of clusters, which in turn depends on the available rate, and I is the total number of iterations needed to converge to an acceptable solution, i.e., when the performance of two consecutive solutions is within a sensitivity threshold ϵ. Specifically, for our simulations reported in Sec. 3.2.5, we chose the Lloyd's algorithm [168], and we notice that ϵ sensitivity target is always achieved within 10 iterations, i.e., $I < 10$.

Policy Compression, Information Bottleneck and Maximum Entropy Reinforcement Learning

We discuss here the connection of the compression schemes presented above with two popular training strategies in RL: the *information bottleneck (IB)* approach [238], and *maximum entropy reinforcement learning (MERL)* [101].

The IB method is a popular way to train an RL agent to maximize a target cumulative reward, with an additional regularization term which accounts for mutual information $I(S;A)$ between the states and actions. As done in [98, 121, 86], the target function $J(\pi)$ to be maximized with respect to policy π is

$$J^{IB}(\pi) = \mathbb{E}\left[R^{\pi}\right] - \beta I(S;A) = \mathbb{E}\left[R_{\pi}\right] - \beta H(A) + \beta H(A|S), \tag{3.16}$$

inducing the policy π to forget task-independent state information, achieving a better generalization performance. A different but similar concept is that of MERL, in which the RL agent is trained to maximize the reward, together with a regularization term that is the entropy of the conditional policy given the state [101, 152], i.e.,

$$J^{MERL}(\pi) = \mathbb{E}\left[R^{\pi}\right] + \beta H(A|S), \tag{3.17}$$

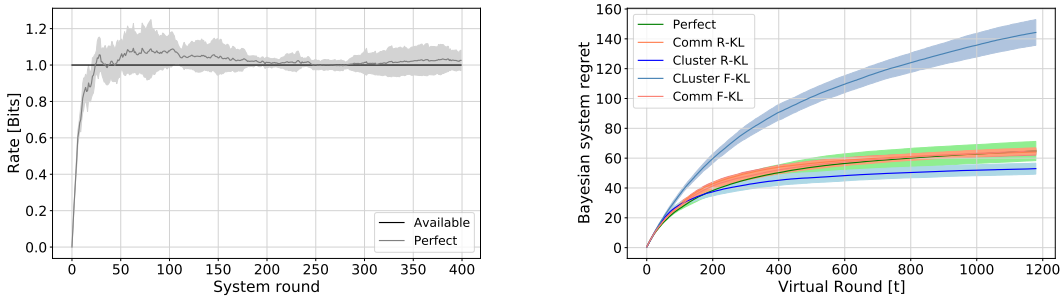
inducing the policy π to more exploration. As proved in [152], MERL naturally appears in the *Control as Inference* framework, in which a probabilistic view of the RL problem is adopted. Specifically, when using a uniform prior distribution over the actions, inferring the optimal posteriors leads to the optimization object in Eq. (3.17). If we now write the distortion-rate function of the RC-CMAB problem, and solve it using the Lagrangian multiplier method, we end up minimizing the objective

$$\mathcal{L}^{RC-CMAB}(Q) = d(Q_{SA}, \pi_{SA}) + \lambda I(S;A), \tag{3.18}$$

where the term $d(Q_{SA}, \pi_{SA})$ plays the role of $\mathbb{E}\left[R^{\pi}\right]$, i.e., it is related to reward maximization by playing according to the optimized posterior, and $\lambda I(S;A)$ is implied by the constraint on the rate. However the difference is that, in our analysis, the parameter λ is optimized to meet the rate constraint in each round, and so it is determined by the problem, rather than being a hyper-parameter to be fine tuned as in the other two formulations. Moreover, in Section 3.2.3 we proved that, if λ is such that $R_{Q_\lambda} = I_{Q_\lambda}(S;A) < R_{\pi^*} = H(A^*)$, then the learning algorithm cannot achieve sub-linear regret. Consequently, it is important to carefully tune the coefficient β when optimizing the policy with the IB and MERL methods. This connection will be also highlighted by the results of the experiments in Section 3.2.5.

3.2.5 Numerical results

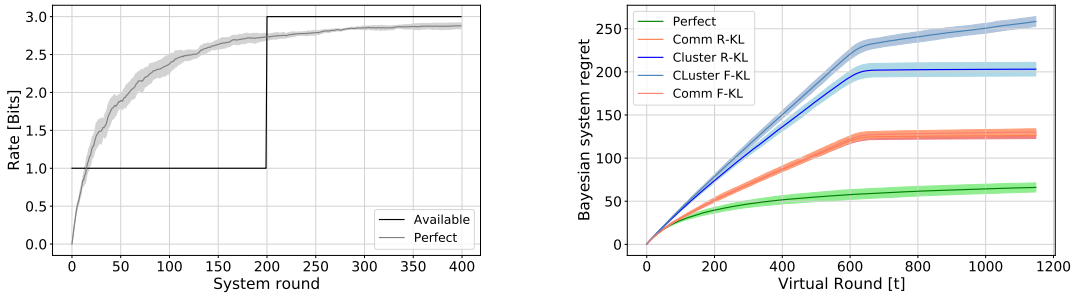
We now analyze the RC-CMAB problem presented in Sec. 3.2.2, and apply both the theoretical and practical policy compression schemes to solve it. In particular, we compare the performance



(a) (b)

Figure 3.2: Rate R_π needed to reliably transmit the policy π as a function of the system round j, together with the imposed rate R (a); average state regret obtained by the different algorithms as a function of the virtual round t (b). In this case, $G = 8$. Curves indicates average rewards $\pm$ one standard deviation over 5 runs.

of the *Perfect* agent, which applies TS without any rate constraint, and thus admits samples from the true posterior π, with the performance of the *rate-constrained* algorithms *Comm R-KL*, *Comm F-KL*, *Cluster R-KL*, and the *Cluster F-KL* agents. The *rate-constrained* agents adopt TS at the decision-maker and, when the rate is not sufficient, transmit a compressed policy Q with the different schemes. Specifically, the *Comm R-KL* agent uses the optimal scheme with reverse KL divergence, and the *Comm F-KL* agent uses the forward KL divergence, as explained in Sec. 3.2.4; the *Cluster R-KL* and *Cluster F-KL* agents implement the practical coding scheme also provided in Section 3.2.4. In every experiment, the context distribution P_S is uniform over the $S = 16$ contexts, and there are $K = 16$ feasible arms, $N = 50$ agents, and the total number of system rounds is $J = 400$. In all the experiments the environment is such that, for each context $s \in \mathcal{S} = \{0, \dots, 15\}$, the best average reward is given by the arm $a \in \mathcal{A} = \{0, \dots, 15\}$ such that $a = \lfloor \frac{s}{G} \rfloor$, where G is an experiment parameter. In particular, the reward behind arm a with context s is a Bernoulli random variable with parameter $\mu(s, a) = 0.8$ if $a = \lfloor \frac{s}{G} \rfloor$, and $\mu(s, a)$ is sampled uniformly in $[0, 0.65]$ otherwise. The average rewards for sub-optimal arms are randomly generated at the beginning of each experiment's run. This set of parameters allows us to study the performance of the different compression schemes, as the degree of correlation between the optimal action distributions and contexts varies. We do not expect notable changes by varying the number of arms, states, and/or reward distributions. For example, increasing the number of arms, or decreasing the gaps between optimal and sub-optimal arms' average rewards, would lead to longer training processes for any algorithms [148], without affecting the performance of compressed policies in comparison to the uncompressed ones. The code to reproduce all the experiments is publicly available on GitHub.[2]



(a) (b)

Figure 3.3: Rate R_π needed to reliably transmit the policy π as a function of the system round j, together with the imposed rate R (a); average state regret obtained by the different algorithms as a function of the virtual round t (b). In this case, $G = 2$. Curves indicates average rewards $\pm$ one standard deviation over 5 runs.

Optimal Rate Constraint

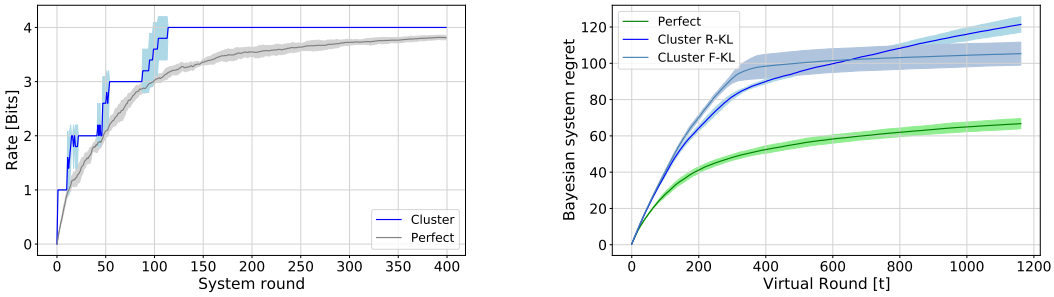
In the first experiment, we set $G = 8$, meaning that the arm with the highest expected reward is $a = 0$ in the first 8 contexts, and $a = 1$ with $s \in \{8, \dots, 15\}$. Consequently, the rate of the optimal policy is $R_{\pi^*} = H(A^*) = 1$ bit, given that P_S is uniform. The maximum available rate R is set to 1 bit for all the agents but the *Perfect* one. Specifically, the agents *Comm R-KL* and *Comm F-KL* adopt a rate $R_{Comm} = \min\{1, R_\pi\}$. Whenever $R_\pi > 1$, the compression schemes explained in Sec. 3.2.4 are adopted. The cluster agents use 1 bit in all the rounds, whereas the *Perfect* agent can always use R_π. The results are presented in Fig. 3.2.

As we can see from Fig. 3.2a, as the learning process goes on, the required rate to transmit the TS policy increases, as the mutual information between S and A increases. Moreover, the rate R_π converges to 1. However, it is interesting to observe Fig. 3.2b, that reports the average state regret as a function of the virtual round index $t = 1, \dots, JN$. The agents *Comm R-KL*, *Comm F-KL* and *Perfect* all achieve similar performance. However, agent *Cluster F-KL* is not able to converge to the optimal policy. Surprisingly, the best performing agent is the more practical *Cluster R-KL*. We argue that this is because it is the agent that makes better use of the IB trick, as 16 different policies are not necessary to represent the optimal responses, given that 2 different distributions are sufficient. We show thus that, when the hyper-parameter β in Eq. (3.16) is optimally tuned, i.e., such that $I(S;A) = H(A^*)$, it is possible to gain in performance when adding the regularization term.

Training with Rate Constraint

In this second experiment, we set $G = 2$, and $R = 2$ for the first 200 system rounds, and $R = 3$ for the remaining 200 rounds. Given $G = 2$, we now have $R_{\pi^*} = 3$ bits, and so the constraint can potentially damage the training process of the rate-constrained agents. As we can see from

[2]https://github.com/FrancescoPase/rccmab



(a) (b)

Figure 3.4: Rate R_π needed to reliably transmit the policy π as a function of the system round j, together with the imposed rate R (a); average state regret obtained by *Perfect* and *Cluster* algorithms, as a function of the virtual round t (b). In this case, $G = 1$. Curves indicates average rewards $\pm$ one standard deviation over 5 runs.

Fig. 3.3b, the *Perfect* agent can easily converge obtaining sub-linear regret. The regret of the rate-constrained agents grows linearly as their rate is imposed to $R = 1$. However, we can see that the two agents that theoretically-optimally trade rate with policy distortion, present the smallest slope in the regret curve. Again, the *Cluster R-KL* agent outperforms *Cluster F-KL*, and achieves almost zero per-round regret as soon as R jumps to 3, meaning that the posteriors converged to the optimal ones even when sampling with $R = 1$. This is not true for *Cluster F-KL*, as it presents larger slope in the regret curve, and the regret keeps growing even when $R = 3$. These observations are consistent with the analysis in [202].

Practical Coding Scheme

This last experiment presents the effectiveness of the practical coding schemes described in Sec. 3.2.4, in the most complex case $G = 1$, i.e., the association between context and optimal arm is a one-to-one map, and the optimal policy π^* needs $R_{\pi^*} = 4$, which is also equal to the context entropy. Consequently, in this case the maximum-complexity policy, i.e., a different arm distribution for each context, is needed [146], and so the IB should be carefully used, and $\beta = 0$ in the end is needed, as no compression scheme can achieve sub-linear regret with $R < 4$ bits. In this case, the number of bits the *Cluster* agents can use is set to $R_{Cluster} = \lceil R_\pi \rceil$. What is interesting to notice here is that, in the long run, the *Cluster F-KL* agent outperforms *Cluster R-KL*, as it is able to converge to the optimal scheme by introducing more exploration. Indeed, even if in the first 400 virtual rounds the *Cluster R-KL* regret curve is below that for *Cluster F-KL*, from $t \sim 600$ the *Cluster F-KL* agent presents better performance. The reason is that, in this case, more exploration should be encouraged in the first steps to discover optimal policies, as done in MERL, by sampling sub-optimal arms more frequently.

3.2.6 Conclusion

In this work, we have proposed and studied a novel rate-constrained remote RL problem, called RC-CMAB. We first proved the performance of the TS strategy when no constraint on the rate is imposed, and the information-theoretic limit for the rate needed to achieve sub-linear regret. We then studied schemes to compress the decision-maker's policy whenever the available rate is not sufficient to reliably transmit the intended actions to the controller. We considered α-divergence as the distortion metric in the rate-distortion problem, and provided the shape of the policies that theoretically optimize the rate-distortion trade-off in close-form, in the cases $\alpha \to 0$, and $\alpha \to 1$, which lead to the reverse and forward KL divergences, respectively. We further proposed a practical compression scheme that relies on the idea of context clustering, and can be adopted to minimize the two analyzed divergence functions. The numerical results confirmed the limit on the achievable rate, and the performance gain that can be achieved when using proper compression strategies in the rate-constrained cases. We further connected and discussed with experiments the relation between our policy compression schemes and the *information bottleneck* approach in RL.

Future steps include the adoption of more advanced practical compression algorithms that reduce the gap between the cluster and optimal schemes. Moreover, the extension of the problem in the context of more general remote RL problems, where the next state depends on the current state and the action taken, is an interesting and challenging direction, as modern algorithms involve the parallelization of the training process exploiting a multitude of agents interacting with many replicas of the same environment.

3.3 Effective Communication in Distributed Reinforcement Learning

Having solved the rate-constrained contextual multi-armed bandit (RC-CMAB) problem, we now move to investigate the role of information in the newly defined remote Partially Observable Markov Decision Process (POMDP), in which Deep Reinforcement Learning (DRL) is used to solve a more complex decision process. To this end, we propose a more practical scheme to compress complex observations and communicate them through the network taking from concepts in the semantic communications literature.

3.3.1 Introduction

The rise of communication metrics that take the content of the message into account, such as the Value of Information (VoI) [272], represents an attempt to approach the problem in practical scenarios, and analytical studies have exploited information theory to define a semantic accuracy metric and minimize distortion [224]. In particular, *information bottleneck* theory [41] has been widely used to characterize Level B optimization [223]. However, translating a practical system model into a semantic space is a non-trivial issue, and the semantic problem is a subject of active research [242, 198]. The effectiveness problem is even more complex, as it implicitly depends on estimating the effect of communication distortion on the control policy and, consequently, on its performance [241]. While the effect of simple scheduling policies is relatively easy to compute [136], and linear control systems can be optimized explicitly [278], realistic control tasks are highly complex, complicating an analytical approach to the Level C problem. Pure learning-based solutions that consider communication as an action in a multi-agent DRL problem, such as emergent communication, also have limitations [91], as they can only deal with very simple scenarios due to significant convergence and training issues. In some cases, the information bottleneck approach can also be exploited to determine state importance [98], but the existing literature on optimizing Level C communication is very sparse, and limited to simpler scenarios [187].

In this work, we consider a dual model which combines concepts from DRL and semantic source coding: we consider an ensemble of Vector Quantized Variational Autoencoder (VQ-VAE) models [244], each of which learns to represent observations using a different codebook. A DRL agent can then select the codebook to be used for each transmission, controlling the trade-off between accuracy and compression. Depending on the task of the receiver, the reward to the DRL agent can be tuned to solve the Level A, B, and C problems, optimizing the performance for each specific task. In order to test the performance of the proposed framework, we consider the well-known CartPole problem, whose state can be easily converted into a semantic one, as its dynamics depend on a limited set of physical quantities. The main contributions of this work are then given by the following:

- We model a remote-control system as a remote POMDP problem and present an efficient solution for learning effective communication through the dynamic compression of learnable features;

- We show that dynamic codebook selection outperforms static strategies for all three levels, and that considering the Level C task can significantly improve the control performance without increasing the bitrate;

- We adopt an explainability framework to understand the choices of the agent in this simple problem, and verify that the Level C dynamic compression captures the receiver's uncertainty in the state estimation and its impact on the expected reward, transmitting only when necessary.

The results and policy analysis lead to significant insights for the design of communication strategies for remote control.

3.3.2 System Model

The recent interest in semantic and effective communications from the research community has driven the development of a wide array of models and conceptualizations, as highlighted in the previous section. At the highest level of abstraction, our purpose is to define a model in which effective communication is meaningful, and the differences between the three problems in Weaver's formulation become clear.

Let us then consider a simple example: we have a remote actuator performing a control task, while a camera observes the results and transmits its observation through a wireless channel. The actuator might have its own sensors, but it relies on the video feed to improve its performance and maintain stable and efficient control. The classical, Level A approach to the problem would be to compress the video as efficiently as possible, minimizing the reconstruction error on frames by using an appropriate codec.The difference between Level A and Level B solutions is then obvious: the former encodes new frames so that the reconstruction fidelity is preserved, while the latter maps elements in the frame to their importance when estimating the physical state of the system. In some control applications, the state can be defined trivially, while in others it may be more complex, but in general, the translation of the video to the state space is unaffected by irrelevant information (such as, e.g., movements in the background).

If we consider Level C, we target control performance directly, and thus further restrict the definition of relevant information: while Level B concerns itself with estimating the system state correctly, a Level C solution only considers errors in the state estimation when they cause performance drops. If the control action is the same over a wide set of states, accuracy then becomes unnecessary, as the actuator only needs a rough estimate of the state to decide what to do; the same happens if there are multiple actions with almost equivalent performance, i.e., if the optimality gap caused by imperfect information remains small.

These natural observations represent the core concepts of effective communication, but implementing them in practical systems is often complex as actions have long-term consequences, and state estimates are based on a history of observations, so that transmitting a message may affect future performance in complex ways. In the following, we provide an analytical framework using the *remote POMDP* approach to objectively evaluate these choices and implement a solution for ef-

Table 3.1: Main notation and definitions.

Symbol	Definition
$\mathcal{S}$	Set of system states
$\mathcal{A}$	Set of feasible actions
$\mathcal{O}$	Set of observations
P	State transition probability function
ω	Observation function
R	Reward function
γ	Discount factor
h	History of stochastic observations
$h^{(r)}$	History of received messages at the robot
π	Policy
G	Expected cumulative discounted reward
$\Phi(\cdot)$	Space of probability distributions over a set
ξ	Belief distribution over the states space $\mathcal{S}$
$\xi^{(o)}$	Belief distribution at the observer
$\xi^{\text{pri},(r)}$	Prior belief distribution at the robot
$\xi^{(r)}$	Belief distribution at the robot
$\mathcal{M}$	Set of messages
Λ	Encoding function
m	Message communicated
$\ell(m)$	Length of message m
ζ	Vector quantizer
$\mathcal{P}$	Picture space
β	Communication cost

fective communication in cyber-physical systems. In Section 3.4.2 we also provide an *information bottleneck* perspective of the problem.

We will denote random variables with capital letters, their possible values with lower-case letters, and sets with calligraphic or Greek capitals. Table 3.1 reports the main symbols we introduce in the following sections for the reader's convenience.

POMDP Definition and Solution

In the standard POMDP formulation [131], one agent needs to optimally control a stochastic process defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, \omega, R, \gamma \rangle$, where $\mathcal{S}$ represents the set of system states, $\mathcal{A}$ is the set of feasible actions, and $\mathcal{O}$ is the observation set. The function $P : \mathcal{S} \times \mathcal{A} \to \Phi(\mathcal{S})$, where $\Phi(\cdot)$ represents the space of probability distributions over a set, gives the state transition probability function. We denote the conditional probability distribution of the next state, given the current state and the selected action, as $P(s'|s,a) = \Pr[S_{t+1} = s' | S_t = s, A_t = a]$. Then, $\omega : \mathcal{S} \to \Phi(\mathcal{O})$ is an observation function, which provides the conditional probabilities $\omega(o|s) = \Pr[O_t = o | S_t = s]$, i.e., the probability of receiving observation o, given that the system is in state s. Finally, function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides the expected reward received by the agent when taking action a in state s, denoted as $R(s,a)$, and the scalar $\gamma \in [0,1)$ is a discount factor used to compute the long-term

reward. We notice that functions P, R, and ω do not depend on the time instant t, thus focusing on homogeneous Markov processes.

The POMDP proceeds in discrete steps indexed by t: at each step t, the agent can infer the system state s_t only from the partial information given by the history of stochastic observations $h_t = (o_t, o_{t-1}, \dots, o_1) \in \mathcal{O}^t$. Based on these observations, and on its policy $\pi : \mathcal{O}^t \to \Phi(\mathcal{A})$, which outputs a probability distribution over the action space for each possible observation history, the agent interacts with the system by selecting an action $A_t \sim \pi(h_t)$. The sampled action a_t is then performed in the real environment, whose hidden state s_t is unknown to the agent, which then receives a feedback from the environment in the form of a (potentially stochastic) reward r_t, with expected value $R_t = R(s_t, a_t)$.[3] The goal for the agent is then to optimize its policy π to maximize the expected cumulative discounted reward $G = \mathbb{E}\Big[\sum_t \gamma^t R_t\Big]$. Having an optimal policy is equivalent to knowing the optimal state-action values, also known as Q-values, and taking action

$$a_t = \pi(h_t) = \arg\max_{a \in \mathcal{A}} Q(h_t, a),$$

where $Q(h_t, a_t) = \mathbb{E}\sum_{\tau=t}^{\infty} \gamma^{\tau - t} R_\tau | h_t, a_t$ is the expected cumulative reward starting from (h_t, a_t).

However, considering the full history of observations makes the solution highly complex, as the length of h_t is potentially unbounded. We then define an estimator $\xi : \mathcal{O}^t \to \Phi(\mathcal{S})$, which outputs the *a posteriori* belief distribution over the state space. We can then recast the original POMDP as a standard Markov decision process (MDP), whose state space is $\mathcal{S}' = \Phi(\mathcal{S})$, i.e., the space of possible belief distributions. Solving the POMDP in this modified belief space has been proved to be optimal in [227].

The policy over this modified MDP is then $\pi : \mathcal{S}' \to \mathcal{A}$, which can be optimized using standard tools [233]. We can also compute the new transition probability and expected reward as in [227]. Given $\xi_t(s) = \Pr(S_t = s \mid h_t)$ and $A_t = a$, we define the *a priori* belief of the state at time $t+1$ as

$$\begin{aligned} \xi_{t+1}^{\text{pri}}(s | \xi_t, a) &= \Pr[S_{t+1} = s \mid \xi_t, A_t = a] \\ &= \sum_{s' \in \mathcal{S}} \xi_t(s') P(s | s', a). \end{aligned}$$

The *a posteriori* belief ξ_{t+1} can then be obtained by performing a Bayesian update, using the new observation o_{t+1} and applying Bayes' theorem using the *a priori* belief as a prior:

$$\begin{aligned} \xi_{t+1}\left(s | \xi_{t+1}^{\text{pri}}, o\right) &= \Pr[S_{t+1} = s \mid \xi_t, A_t = a, O_{t+1} = o] \\ &= \frac{\omega(o|s)\xi_{t+1}^{\text{pri}}(s)}{\sum_{s' \in \mathcal{S}} \xi_{t+1}^{\text{pri}}(s')\omega(o|s')}. \end{aligned} \tag{3.19}$$

These update equations allow us to compute the modified transition probability matrix P' for the belief MDP, which is then defined by the tuple $\langle \Phi(\mathcal{S}), \mathcal{A}, P', R, \gamma \rangle$.

[3] As the reward signal is only provided during training, it cannot be used to infer the value of s_t.

The Remote POMDP

We consider a variant of the POMDP that we define *remote POMDP*, in which two agents are involved in the process. The first agent, i.e., the *observer*, receives observation $O_t \in \mathcal{O}$, and needs to convey such information to a second agent, i.e., the *robot*, through a constrained communication channel, which limits the number of bits the observer can send. Consequently, the amount of information the observer can send to the robot is limited. The robot then chooses and takes an action in the physical environment. This system can formalize many control problems in future Industrial Internet of Things (IIoT) systems, as sensors and actuators may potentially be geographically distributed, and the amount of information they can exchange to accomplish a task is limited by the shared wireless medium, which has to be allocated to the many devices installed in the factory, as well as by the energy limitations on the sensors. Similar systems have been analyzed in [241, 93]. We will now analyze the problems for the two agents, considering a case in which communication and control are designed separately. Joint control and communication approaches [170] can outperform separate approaches by tuning the two agents' policies to each other, but they introduce additional training complexity, and will not be considered in this work. In the following, we will refer to variable x related to the robot as $x^{(r)}$, while the corresponding variable on the observer side will be denoted by $x^{(o)}$.

The Robot-Side POMDP

We denote the message communicated to the robot at time step t as $m_t \in \mathcal{M}$. The set of possible messages $\mathcal{M}$ forms the set of observations that are available to the robot, and the history of these observations is given by $h_t^{(r)} = \{m_t, \dots, m_1\}$, which is the sequence of messages received up to time t. We can then see the robot as an agent with its own POMDP, in which the observations are filtered by both the partial knowledge of the observer and the further distortion produced by the fact that these observations are encoded and communicated through a constrained channel. The robot-side POMDP is then defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{M}, P, \pi^{(o)}, R, \gamma \rangle$, as observations depend on the observer's policy.

The message transmitted from the observer to the robot modifies the belief distribution over the next state as a Bayesian update. Let us define the distribution over the current state, given that the message m_t has been received, as $\xi_t^{(r)}$. For example, if the communicated message m_t contains the correct state $S_t = s$, the belief distribution becomes deterministic, i.e., $\xi(s' \mid m_t) = \delta_{s,s'}$, where $\delta_{m,n}$ is the Kronecker delta function, equal to 1 if the two arguments are the same and 0 otherwise. Ideally, an intelligent observer will allocate more communication resources and thus provide more precise messages if the *a priori* distribution of the robot is far from the one estimated by the observer. The modified MDP is then defined by the tuple $\langle \Phi(\mathcal{S}), \mathcal{A}, P^{(r)}, R, \gamma \rangle$, where $P^{(r)}$ represents the Bayesian update function. According to the previous notation, we can express the optimal action at time $t+1$ as

$$a_{t+1} = \arg\max_{a \in \mathcal{A}} Q\left(\xi_t^{(r)}, a\right),$$

where $\xi_t^{(r)}$ is the current belief at the robot side (after message m_t is received). The robot's reward is simply given as the reward of the original POMDP, i.e., the control performance in the environment. The optimal policy can be reached by using standard DRL tools.

The Observer-Side POMDP

On the other side, the observer needs to encode its belief $\xi_t^{(o)}$ in a message $m_t \in \mathcal{M}$ and transmit it. We can then consider the observer-side POMDP, in which the action set corresponds to the set of messages $\mathcal{M}$ and the state space is represented by the belief from the observed results. The tuple defining this POMDP is $\langle \mathcal{S}, \mathcal{M}, \mathcal{O}, P, \omega, R^{(o)}, \gamma \rangle$. We assume that the observer knows the robot's policy, i.e., it can know the actions that the robot takes in the environment and use them to improve its estimate of the state. This can also be accomplished if the robot transmits the actions it takes as feedback to the observer. As described above for the robot-side problem, we can transform this POMDP into the belief MDP given by $\langle \Phi(\mathcal{S}) \times \Phi(\mathcal{S}), \mathcal{M}, P^{(o)}, R^{(o)}, \gamma \rangle$, where $P^{(o)}$ is the Bayesian update given in (3.19). We highlight that the observer needs to keep track of both its own and the robot's belief, as the effectiveness of communication depends on the difference between the two, and the state of the observer is given by $\left\langle \xi_t^{(o)}, \xi_t^{\mathrm{pri},(r)} \right\rangle$.

The objective of the observer is to minimize channel usage, i.e., communicate as few bits as possible, while maintaining the highest possible performance in the control task: the expected reward $R^{(o)}$ depends on both components. If it transmits message m, whose length in bits is $\ell(m)$, the observer then gets a penalty $\beta\ell(m)$, where $\beta \in \mathbb{R}^+$ is a cost parameter. In order to optimize its policy, the observer also needs to have a way to gauge the *value* of information, which is a complex problem: information theory, and in particular rate-distortion theory, have provided the fundamental limits when optimizing for the technical problem, i.e., Level A, where the goal is to reconstruct the source signals with the highest fidelity [69]. We will discuss the definition of VoI in the following sections.

As the complexity of the problem is massive, we restrict ourselves to a smaller action space by making a simplifying assumption, which allows us to separate the problem: the observer does not transmit the entire belief distribution, which may be implicit, but rather the observation O_t. We then consider the encoding function $\Lambda : \mathcal{O} \times \Phi(\mathcal{S}) \to \mathcal{M}$, which will generate a message $M_t = \Lambda\left(O_t \mid \xi_t^{(o)}\right)$ to be sent to the robot at each step t.

Observer Reward in Remote POMDPs

The first and simplest way to solve the remote POMDP problem is to blindly apply standard Level A rate-distortion metrics to compress the sensor observations into messages to be sent to the agent. As an example, in the CartPole problem analyzed in this work (see Sec. 4.3.5), one sensor observation is given by two consecutive 2D camera acquisitions. The observer's policy is then independent of the robot's task, and can be computed separately. The Level A reward function $R_A^{(o)}$ is then given as follows:

$$R_A^{(o)}\left(\left\langle \xi_t^{(o)}, \xi_t^{\mathrm{pri},(r)} \right\rangle, m\right) = -d_A(o_t, \hat{o}_t) - \beta\ell(m). \tag{3.20}$$

In the CartPole case, a natural distortion metric is the image Peak Signal to Noise Ratio (PSNR), an image quality metric proportional to the logarithm of the normalized Mean Square Error (MSE) between the images. Naturally, encoding the observation with a higher precision will require more bits, as the set of messages needs to be bigger.

The Level B problem considers the projection of the raw observations into a significantly smaller *semantic space*, over which we measure distortion using function d_B, explicitly capturing the error over the needed physical system information, e.g., the angular position and velocity of the pole in the CartPole problem. The Level B reward function $R_B^{(o)}$ is then given as follows:

$$R_B^{(o)}\left(\left\langle \xi_t^{(o)}, \xi_t^{\mathrm{pri},(r)} \right\rangle, m\right) = -d_B\left(\xi_t^{(o)}, \xi_t^{(r)}\right) - \beta\ell(m). \tag{3.21}$$

In our CartPole case, this may be simply represented by the MSE between the best estimate of the state at the transmitter and receiver.

Finally, we can consider the Level C system. In this case, the distortion metric is not needed, as the control performance can be used directly, and the reward $R_C^{(o)}$ is:

$$R_C^{(o)}\left(\left\langle \xi_t^{(o)}, \xi_t^{\mathrm{pri},(r)} \right\rangle, m\right) = R\left(\xi_t^{(r)}, \pi^{(r)}\left(\xi_t^{(r)}\right)\right) - \beta\ell(m). \tag{3.22}$$

The VoI of message m, $V\left(\xi_t^{\mathrm{pri},(r)}, m\right)$, can then be given by the difference between the expected performance of the robot with this information and without it:

$$V\left(\xi_t^{\mathrm{pri},(r)}, m\right) = Q\left(\xi_t^{(r)}, \pi^{(r)}\left(\xi_t^{(r)}\right)\right) - Q\left(\xi_t^{\mathrm{pri},(r)}, \pi^{(r)}\left(\xi_t^{\mathrm{pri},(r)}\right)\right). \tag{3.23}$$

Thus, the optimal Level C observer policy $\pi_C^{(o)}$ will balance the trade-off between the performance at the receiver and the communication cost not only in the current time step but also in the long term. This foresighted behavior is essential when considering that the belief distributions incorporate the memory of previously received messages. Providing information that does not improve the expected reward in the next step might still be worth the cost if it allows the robot to improve its estimate, reducing the need for future communication.

3.3.3 Proposed Solution

In this section, we introduce the architecture we used to represent Λ, the VQ-VAE, and discuss the remote POMDP solution. As the VQ-VAE model is not adaptive, we consider an ensemble model with different quantization levels, limiting the choice of the observer to *which* VQ-VAE model to use in the transmission. As we mentioned, directly learning the encoding is highly complex, with a vast action space, and techniques such as emergent communication that learn it explicitly are limited to scenarios with very simple tasks and immediate rewards. By restricting the problem to a smaller action space, we may find a slightly suboptimal solution, but we can deal with much more complex problems.

Deep VQ-VAE Encoding

In order to represent the encoding function Λ, and to restrict the observer-side POMDP to a more manageable action space, the observer exploits the VQ-VAE architecture introduced in [244]. The VQ-VAE is built on top of the more common Variational Autoencoder (VAE) model, with the additional feature of finding an optimal discrete representation of the latent space. The VAE is used to reduce the dimensionality of an input vector $X \in \mathbb{R}^I$, by mapping it into a stochastic latent representation $Z \in \mathbb{R}^L \sim q_\nu(Z|X)$, where $L < I$. The stochastic encoding function $q_\nu(Z|X)$ is a parameterized probability distribution represented by a neural network with parameter vector ν. To find optimal latent representations Z, the VAE jointly optimizes a decoding function $p_\theta(\hat{X}|Z)$ that aims to reconstruct X from a sample $\hat{X} \sim p_\theta(\hat{X}|Z)$. This way, the parameter vectors ν and θ are usually jointly optimized to minimize the distortion $d(X, \hat{X})$ between the input and its reconstruction, given the constraint on Z, while reducing the distance between $q_\nu(Z|X)$, and some prior $q(Z)$ [137] used to impose some structure or complexity budget.

However, in practical scenarios, one needs to digitally encode the input X into a discrete latent representation. To do this, the VQ-VAE quantizes the latent space by using N K-dimensional codewords $z_1, \ldots, z_N \in \mathbb{R}^K$, forming a dictionary with N entries. Moreover, to better represent 3D inputs, the VQ-VAE quantizes the latent representation Z using a set of F blocks, each quantizing one feature $f(X)$ of the input, and chosen from a set of N possible codewords. We denote the set containing all the N^F possible concatenated blocks with $\mathcal{M}(N)$, as it represents the set of all possible messages the observer can use to convey to the robot the information on the observation O, by using F discrete N-dimensional features. The peculiarity of the VQ-VAE architecture is that it jointly optimizes the codewords in $\mathcal{M}(N)$ together with the stochastic encoding and decoding functions q_ν and p_θ, instead of simply applying fixed vector quantization on top of learned continuous latent variables Z. When the communication budget is fixed, i.e., the value of L is constant, the protocol to solve the remote POMDP is rather simple: first, the observer trains the VQ-VAE with $N = 2^{F^{-1}L}$ to minimize the technical, semantic, or effective distortion d_α, depending on the problem; then, at each step t, the observer computes $\hat{m} \sim q_\nu(\cdot|o_t)$, and finds $m_t = \text{argmin}_{m \in \mathcal{M}(N)} \|m - \hat{m}\|_2$. The message m_t is sent to the robot, which can optimize its decision accordingly.

Dynamic Feature Compression

We can then consider the architecture shown in Fig. 3.5, consisting of a set of VQ-VAEs $\mathcal{V} = \{\zeta_\varnothing, \zeta_1, \ldots, \zeta_V\}$, where each VQ-VAE ζ_v compresses each feature using v bits. We also include a null action $\zeta_\varnothing$, which corresponds to not transmitting anything. As we only consider the communication side of the problem, the robot is trained beforehand using the messages with the finest-grained quantization, which are compressed with the VQ-VAE ζ_V with the largest codebook. The robot can then perform three different tasks, corresponding to the three communication problems: it can decode the observation (Level A) with the highest possible accuracy, using the decoder part of the VQ-VAE architecture; it can estimate the hidden state (Level B) using a supervised learning solution; or it can perform a control action based on the received information and observe

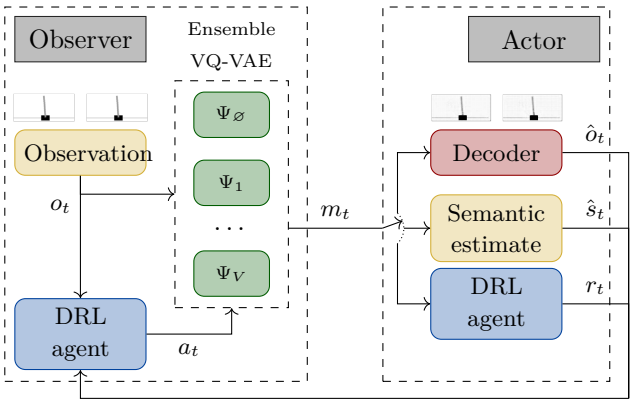


Figure 3.5: Dynamic feature compression architecture.

its effects (Level C).

In all three cases, the dynamic compression is performed by the observer, based on the feedback from the robot. The observer side of the remote POMDP, whose reward is given in (3.20)-(3.22), is restricted to the choice of ζ_v, i.e., to selecting one of the possible codebooks learned by each VQ-VAE in the ensemble model, or to avoid any transmission. As we described in the previous section, the type of reward depends on the communication problem that the observer is trying to solve: at Levels A and B, the observer aims at minimizing distortion in the observation and semantic space, respectively. At Level C, the objective is to maximize the robot's reward.

In all three cases, *memory* is important: representing snapshots of the physical system in consecutive instants, subsequent observations have high correlations, and the robot can glean a significant amount of information from past messages. This is an important advantage of dynamic compression, as it can adapt messages to the estimated knowledge at the receiver side.

While the observer is adapting its transmissions to the robot's task, the robot's algorithms are fixed. They could themselves be adapted to the dynamic compression strategy, but this joint training is significantly more complex, and we consider it as a possible extension of this work.

RL implementation

There are two policies in the considered system, one for the observer and one for the robot and in both cases the policies are learned through the Actor Critic algorithm. This means that an agent learns a parametric policy π_λ and a Q-values estimator. Both the policy and the Q-values are neural networks. In order to take into account the past observations the two networks share a Long Short-Term Memory (LSTM) layer which estimates a latent state which is then given as input to both the policy and the values estimator. This architecture avoids explicitly modeling the belief distribution which may be complicated to treat in continuous settings like the one considered in this work. This practical choice is also useful to avoid decoding the latent features discovered by the VQ-VAE back in the observation space $\mathcal{O}$ or in the physical state space $\mathcal{S}$, increasing the potential for errors. Indeed, the quantized features communicated with the message m_t contain a

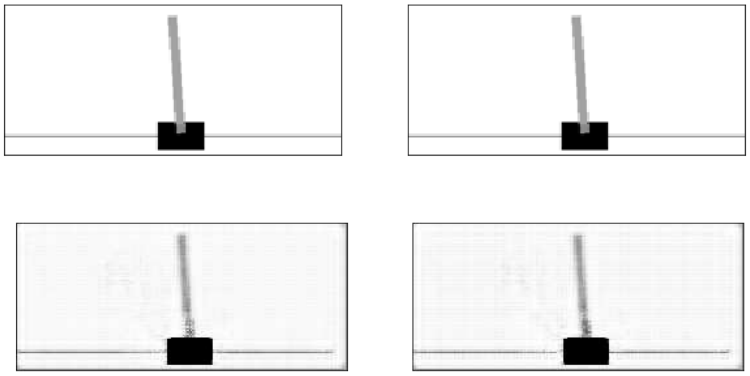


Figure 3.6: Example of the original and reconstructed observation.

structured representation of the observation space which can be used effectively by an LSTM to estimate the *true state*. The training algorithm is the standard Advantage Actor Critic (A2C), but the replay buffer is appropriately modified to take into account the history of previously received messages.

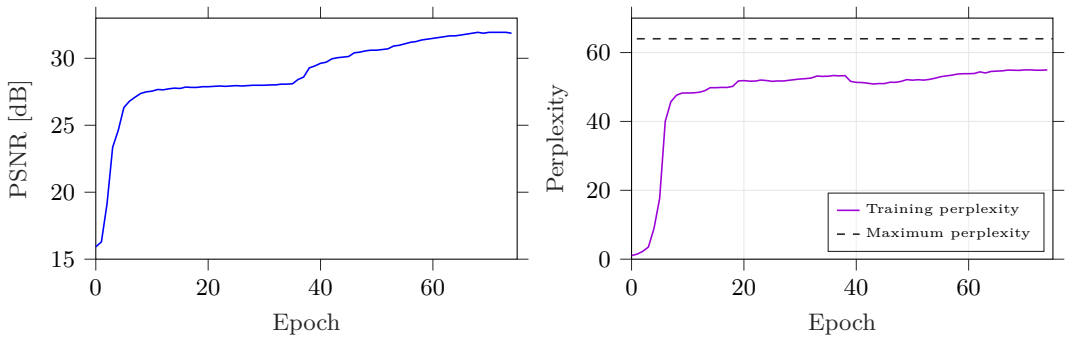
3.3.4 Simulation Settings and Results

The underlying use case analyzed in this work is the well-known CartPole problem, as implemented in the *OpenAI Gym* library.[4] In this problem, a pole is installed on a cart, and the task is to control the cart position and velocity to keep the pole in equilibrium. The physical state of the system is fully described by the cart position x_t and velocity $\dot{x}_t$, and the pole angle ψ_t and angular velocity $\dot{\psi}_t$. Consequently, the true state of the system is $s_t = (x_t, \dot{x}_t, \psi_t, \dot{\psi}_t)$, and the semantic state space is $\mathcal{S} \subset \mathbb{R}^4$ (because of physical constraints, the range of each value does not actually span the whole real line).

At each step t, the observer senses the system by taking a black and white picture of the scene, which is in a space $\mathcal{P} = \{0, \dots, 255\}^{180\times360}$. To take the temporal element into account, an observation O_t includes two subsequent pictures, at times $t-1$ and t, so that the observation space is $\mathcal{O} = \mathcal{P} \times \mathcal{P}$. An example of the transmission process is given in Fig. 3.6, which shows the original version sensed by the observer (above) and the reconstructed version at the receiver (below) when using a trained VQ-VAE with $N_\zeta = 6$.

In the CartPole problem, the action space $\mathcal{A}$ contains just two actions `Left` and `Right`, which push the cart to the left or to the right, respectively. At the end of each step, depending on the true state s_t, and on the taken action a_t, the environment will return a deterministic reward $R_t = -x_{\max}^{-1}|x_t| - \psi_{\max}^{-1}|\psi_t|,\ \forall t$, where $x_{\max} = 4.8$ m and $\psi_{\max} = \frac{2\pi}{15}$ rad (equivalent to 24°) are the maximum values for the two quantities. If the angle or cart position go outside the boundaries, the episode is over, and the agents do not accumulate any more reward. The goal for the two

[4] `https://www.gymlibrary.dev/environments/classic_control/cart_pole/`



(a) PSNR.

(b) Perplexity.

Figure 3.7: Training of the VQ-VAE model with $N_\zeta = 6$.

agents is thus to maximize the cumulative discounted sum of the reward R_t, while limiting the communication cost.

The Coding and Decoding Functions

As mentioned in Sec. 3.3.2, the observer can optimize its coding function Λ according to different criteria depending on the considered communication problem. However, as we explained in Sec. 3.3.3, optimizing Λ without any parameters is usually not feasible due to the curse of dimensionality on the action space. Consequently, we rely on a pre-trained set $\mathcal{V}$ of VQ-VAE models, whose codebooks are optimized to solve the technical problem, i.e., minimizing the distortion on the observation measured using the MSE: $d_A(o, \hat{o}) = \text{MSE}(o, \hat{o})$. The training performance of the VQ-VAE with $N_\zeta = 6$ is shown in Fig. 3.7: the encoder converges to a good reconstruction performance, which can be measured by its *perplexity*. The perplexity is simply $2^{H(p)}$, where $H(p)$ is the entropy of the codeword selection frequency, and a perplexity equal to the number of codewords is the theoretical limit, which is only reached if all codewords are selected with the same probability. The perplexity at convergence is 54.97, which is close to the theoretical limit for a real application.

The observer then uses DRL to foresightedly optimize the quantization level $N_\zeta(t)$ at each time step, maximizing the expected long-term reward for each communication problem. We train the observer to solve the level-specific coding problem by designing three different rewards, depending on the considered communication level:

1. *Level A (technical problem):* The distortion metric for the observer is $d_A(o_t, \hat{o}_t) = \text{PSNR}(o_t, \hat{o_t}) - \beta\ell(m_t)$, as part of the reward definition from (3.20). The PSNR is an image fidelity measure proportional to the logarithm of the normalized MSE between the original and reconstructed image;

2. *Level B (semantic problem):* The distortion metric is $d_B(\hat{s}_t^{(o)}, \hat{s}_t^{(r)}) = -\text{MSE}(\hat{s}_t^{(o)}, \hat{s}_t^{(r)})$, as part of the reward defined in (3.21), and the decoder needs to estimate the underlying physical state s_t by minimizing the MSE, i.e., the distance between $\hat{s}_t^{(o)}$ and $\hat{s}_t^{(r)}$ in the

Table 3.2: Simulation Parameters.

Parameter	Value	Description
$H \times W$	160×360	Image size
V	7	Number of quantizers
F	8	Number of latent features
D_{emb}	8	Embedding dimension of features
B	256	Batch size
γ	0.95	Discount factor
T	500	Maximum number of steps for an episode
α_{enc}	10^{-3}	VQ-VAE learning rate
α_{Reg}	10^{-4}	Regressor learning rate
α_{A2C}	10^{-4}	A2C learning rate
D	5×10^4	Size of the VQ-VAE training dataset
N_{enc}	100	Encoder training epochs
N_{rob}	2×10^4	Robot policy training episodes
N_{obs}	1×10^5	Observer policy training episodes
N_{test}	1000	Number of test episodes

semantic space. In our case, the estimator used to obtain the estimates is a pre-trained supervised LSTM neural network;

3. *Level C (effective problem):* In this case, there is no direct distortion metric, and the control performance is used directly as in (3.22). The policy $\pi^{(r)}$ is given by an actor-critic agent implementing an LSTM architecture, pre-trained using data with the highest available message quality (6 bits per feature).

We observe that, in this case, $i(s) = s$, i.e., the task depends on all the semantic features contained in S_t. However, the 4 components of the state do not carry the same amount of information to the robot: depending on the system conditions, i.e., the state S_t, some pieces of information are more relevant than others.

Neural Network Architecture and Training

The VQ-VAE architecture is made with Convolutional Neural Network (CNN) layers to extract latent features and it is trained separately before the training of the control policy. To this end, a dataset of observations is collected through a random policy. After that, we train an encoding network, the vector quantization layer and the decoder jointly as in the standard VQ-VAE [244]. The first vector quantization layer learned contains the highest number of codewords. After that, we fix the encoder and the decoder and just train the other vector quantization layers, obtaining multiple quantizers over the same latent space discovered by a common encoder. The hyperparameters used to train the VQ-VAE are reported in Table 3.2. After obtaining the V quantizers, we train the policy using the standard A2C algorithm. Table 3.3 shows the Encoder-Decoder layers of the VQ-VAE. In Table 3.4, the layers of the implemented Regressor and the Actor-critic neural networks are reported. All the neural network (NN)s are implemented through

Table 3.3: Encoder-Decoder parameters.

Layer type	Size	Kernel size	Stride
	Encoder		
`Conv2d + ReLU`	64	10×11	8×9
`Conv2d + ReLU`	64	12×12	10×10
`Conv2d + ReLU`	128	3×3	1×1
`ResidualStack`	2	3×3	1×1
`Conv2d`	8	3×3	1×1
	Decoder		
`ResidualStack`	2	3×3	1×1
`Conv2d + ReLU`	128	3×3	1×1
`Conv2d + ReLU`	64	12×12	10×10
`Conv2d`	64	10×11	8×9

Table 3.4: Recurrent architectures.

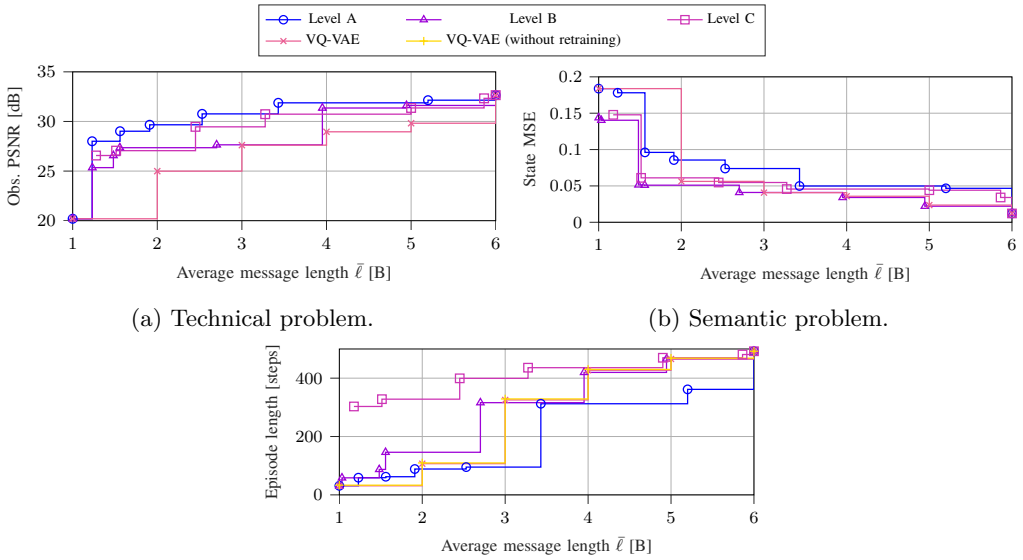
Layer type	Inputs	Outputs	Description
	Regressor		
`LSTM + ReLU`	64	64	Single recurrent layer
`Linear + ReLU`	64	128	Hidden layer
`Linear`	128	1	Output layer
	Actor-critic		
`LSTM + ReLU`	64	64	Single recurrent layer
`Linear + ReLU`	64	128	Hidden layer
`Linear`	128	1	Output layer (Value)
`Linear + softmax`	128	$\lvert\mathcal{A}\rvert$	Output layer (Policy)

the Pytorch labrary. Once the robot policy has been obtained, we can train the observer policy. The observer learns a policy through the same A2C algorithm, but in this case the input to the policy are the features before quantization. A unique observer policy is trained for different values of the trade-off parameter β and for different communication levels. For further details on the implementation, training and testing process, we refer to the publicly available simulation code.[5]

Results

We assess the performance of the three different tasks in the CartPole scenario by simulation, measuring the results over 1000 episodes after convergence. Fig. 3.8 shows the performance of the various schemes over the three problems, compared with a static VQ-VAE solution with a constant compression level. In the Level C evaluation, we also consider a static VQ-VAE solution in which the robot is not retrained for each N_ζ, but is only trained for $N_\zeta = 6$ (i.e., 48 bits per message) as for the dynamic scheme. We trained the dynamic schemes with different levels

[5] `https://www.github.com/pietro-talli/tmlcn_code`



(a) Technical problem.

(b) Semantic problem.

(c) Effective problem.

Figure 3.8: Performance of the communication schemes on the three levels of the remote POMDP.

of the communication cost β, so as to provide a full picture of the adaptation to the trade-off between performance and compression. We also introduce the notion of *Pareto dominance*: an n-dimensional tuple $\eta = (\eta_1, \ldots, \eta_n)$ Pareto dominates η' (which we denote as $\eta \succ \eta'$) if:

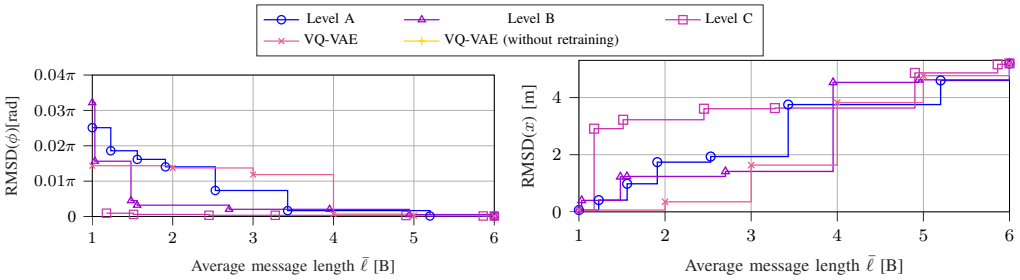
$$\eta \succ \eta' \iff \exists i : \eta_i > \eta'_i \wedge \eta_j \geq \eta'_j \, \forall j. \tag{3.24}$$

We can extend this to schemes with multiple possible configurations. The definition of Pareto dominance for schemes x and y is: $x \succ y \iff \exists \eta_x \succ \eta_y \, \forall \eta_y$, i.e., for each configuration of scheme y, there is a setting of x that Pareto dominates it. In other words, we can always tune scheme x so that it outperforms any configuration of scheme y on all metrics.

We first consider the technical problem performance, shown in Fig. 3.8a: as expected, the Level A dynamic compression outperforms all other solutions, and its performance is Pareto dominant with respect to static compression. Interestingly, the Level B and Level C solutions perform worse than static compression: by concentrating on features in the semantic space or the task space, these solutions remove information that could be useful to reconstruct the full observation, but is meaningless for the specified task.

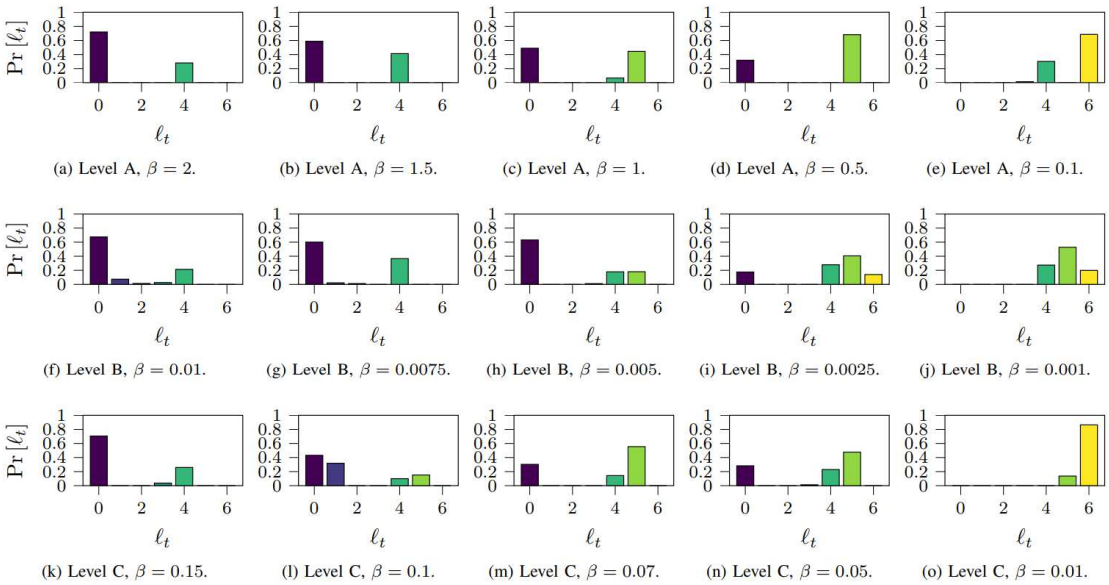
In the semantic problem, shown in Fig. 3.8b, a lower MSE on the reconstructed state is better, and the Level B solution is Pareto dominant with respect to all others. The Level A solution also Pareto dominates static compression, while the Level C solution only outperforms it for higher compression levels, i.e., on the left side of the graph.

Finally, Fig. 3.8c shows the performance at the effectiveness level, summarized by how long the CartPole system manages to remain within the position and angle limits. The Level C solution significantly outperforms all others, but is not strictly Pareto dominant: when the communication



(a) Angular RMSD from the central position. (b) Position RMSD from the central position.

Figure 3.9: Other performance metrics relative to the CartPole control problem.



(a) Level A, $\beta = 2$. (b) Level A, $\beta = 1.5$. (c) Level A, $\beta = 1$. (d) Level A, $\beta = 0.5$. (e) Level A, $\beta = 0.1$.

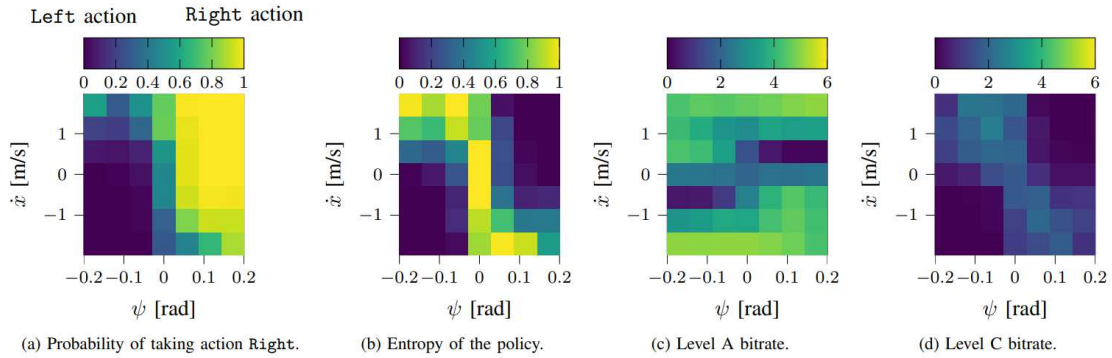
(f) Level B, $\beta = 0.01$. (g) Level B, $\beta = 0.0075$. (h) Level B, $\beta = 0.005$. (i) Level B, $\beta = 0.0025$. (j) Level B, $\beta = 0.001$.

(k) Level C, $\beta = 0.15$. (l) Level C, $\beta = 0.1$. (m) Level C, $\beta = 0.07$. (n) Level C, $\beta = 0.05$. (o) Level C, $\beta = 0.01$.

Figure 3.10: Distribution of the selected compression levels.

constraint is very tight, setting a static compression and retraining the robot to deal with the specific VQ-VAE used may provide a slight performance advantage. In general, almost perfect control can be achieved with less than half of the average bitrate of the static compressor, which can only reach similar levels of performance at a much higher communication cost. We also note that, in this case, the Level B solution performs worst: choosing the solution that minimizes the semantic distortion might not be matched to the task, as the state variables have equal weight, while a higher precision might be required when the quantization error might change the robot's action.

Another analysis is conducted on the way the CartPole is controlled with the different communication policies. Fig. 3.9 shows the Angular Root Mean Squared Deviation (RMSD) (Fig. 3.9a)



(a) Probability of taking action `Right`. (b) Entropy of the policy. (c) Level A bitrate. (d) Level C bitrate.

Figure 3.11: Analysis of the transmission policy as a function of the pole angle and cart linear velocity $\dot{x}$.

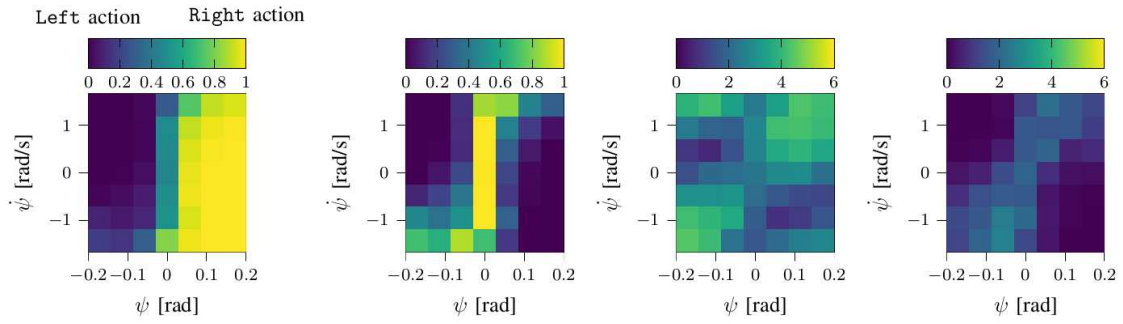
and the Position RMSD (Fig. 3.9b), defined as:

$$\mathrm{RMSD}(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x_{\mathrm{target}})^2},$$

where x_{target} is the desired value of the controlled process and x_i is the recorded process at time step i. Both RMSD are computed with respect to the central and vertical position of the CartPole: $x_{\mathrm{target}} = 0$ and $\phi_{\mathrm{target}} = 0$. These results help to evaluate how well the control dynamics keep the CartPole near the optimal central position and to assess the smoothness of the resulting pole oscillations. It is possible to see that in general, a higher rate allows to keep the angular RMSD smaller. In particular, in the Level C system, the values are the smallest. However, this comes at the cost of deviating more from the central position, as shown in the figure. The policy prioritizes the stabilization of the pole oscillations, though this requires deviating from the central position. This is because swings in the pole's angle are harder to control due to the instability of inverted pendulum, and there is a significant risk that the pole might go out of the acceptable range, ending the episode.

Analysis of the communication policy

We can then use an explainability approach to gain further insights on how effective communication operates. Fig. 3.10 shows the distribution of the quantization level selected by an observer trained for the three different communication levels. We note that the scale of β is different, as the reward process takes values in different ranges (e.g., the PSNR is in dB while the reward of the MDP is between -1 and 1), but the resulting bitrates are similar. The similarity in the compression level distributions at the three levels are striking. For lower values of β, the observer uses the action $\Phi_{\varnothing}$, which corresponds to no transmission, more often. As β decreases, the communication cost becomes lower, and thus the observer chooses longer messages more often. Another common feature is that quantizing features using $1, 2$ or 3 bits is a rare choice. This shows that the memory



(a) Probability of taking action `Right`. (b) Entropy of the policy. (c) Level A bitrate. (d) Level C bitrate.

Figure 3.12: Analysis of the transmission policy as a function of the pole angle and angular velocity $\dot{\psi}$.

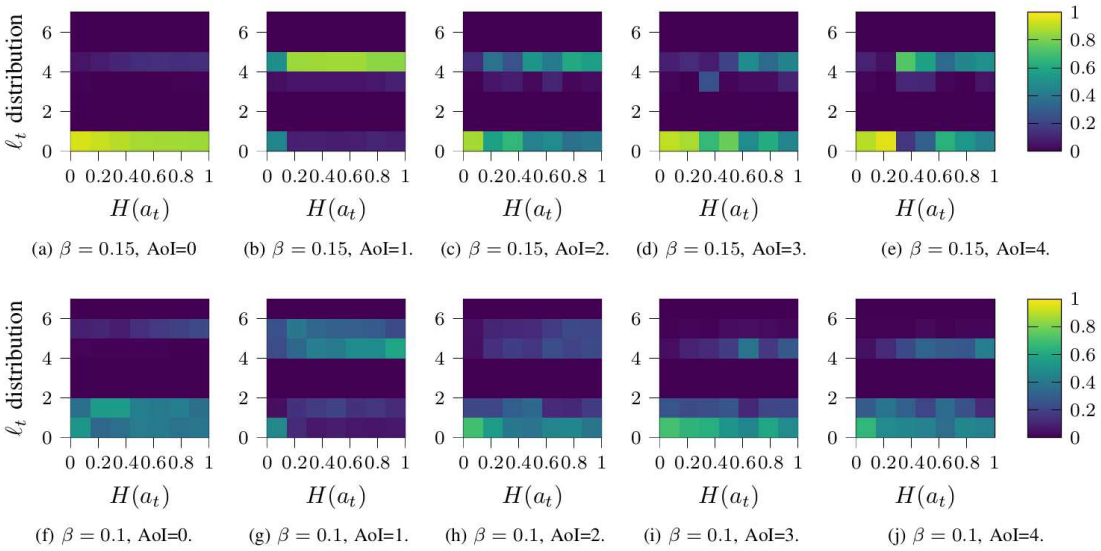
implemented implicitly in the system through the LSTM is powerful enough to obtain adequate beliefs based on past messages, so that the observer can rely on it and not send anything, avoiding the cost of transmitting even a roughly quantized update and rather transmitting intermittent updates at a higher quality.

However, the real difference between the three policies is given by *when* they decide not to transmit. Therefore, we propose an analysis based on the visualization of the observer policy and the receiver policy. In Fig. 3.11, four colormaps show different policies projected in the same domain: the pole angle on the x-axis and the cart velocity on the y-axis. More specifically, we quantize the projected state into cells and show the policy of the robot and the observer in each cell. Fig. 3.11 (a) shows the robot actions, averaged among 10^6 samples. Since in the CartPole problem the actions are binary, we represent the probability of choosing action `Right` in a range between 0 and 1. Fig. 3.11 (b) depicts the entropy of the robot policy. We considered the action probability in the previous figure and compute the action entropy as follows:

$$H(a) = -\sum_{a \in \{0,1\}} p(a) \log_2 \left(p(a)\right),$$

where $p(a)$ is empirically estimated by counting the number of times each action is chosen when the state is in the projected cell. Fig. 3.11 (c) and Fig. 3.11 (d) show the average number of bits transmitted in each cell when optimizing for level A and C, respectively. This can be seen as the average number of bits that the transmitter allocates for each projected slice of the state space. Fig. 3.12 shows the same results but for a different physical state projection, mapping the angle ϕ on the x-axis and the pole angular velocity $\dot{\phi}$ on the y-axis.

In both figures, there is a strong correspondence between the states where the robot entropy is higher and the states where the level C policy allocates a higher number of bits. This confirms that an effective observer policy manages to discriminate the uncertainty at the robot side. In regions of the state space where it is more difficult to retrieve the correct action, i.e., the action entropy is higher, the observer will provide the robot with more precise information by sending



(a) $\beta = 0.15$, AoI=0 (b) $\beta = 0.15$, AoI=1. (c) $\beta = 0.15$, AoI=2. (d) $\beta = 0.15$, AoI=3. (e) $\beta = 0.15$, AoI=4.

(f) $\beta = 0.1$, AoI=0. (g) $\beta = 0.1$, AoI=1. (h) $\beta = 0.1$, AoI=2. (i) $\beta = 0.1$, AoI=3. (j) $\beta = 0.1$, AoI=4.

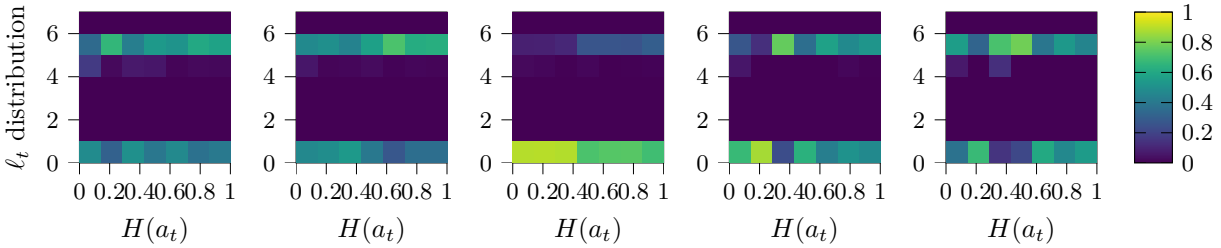
Figure 3.13: Level C observer action distribution for different robot action entropy levels and values of β.

longer messages. There are regions where the robot action is always the same, e.g., whenever the cart is moving fast and the tip of the pole is pointing to the same side the cart is moving towards. In these cases, the entropy is extremely low, and the transmitter can avoid sending new updates to the robot. This is due to the fact that, even if the estimated state at the receiver differs from the observed one, the action to perform remains the same and will be to push further the cart to try to get the pole more vertical. Recalling (3.23), we note that if $Q\left(\xi_t^{(r)}, \pi^{(r)}\left(\xi_t^{(r)}\right)\right)$ is very sensitive to small variations in $\xi_t^{(r)}$, then the gap in (3.23) is going to be significant, leading the observer to choose to send precise information. In principle, a level C transmitter could reduce the message length or even avoid transmission as long as the robot is able to choose the correct actions, even though its belief is incorrect. An optimal communication scheme approximately follows

$$\ell_t \propto V\left(\xi_t^{\mathrm{pri},(r)}, m\right),$$

which means that the message length is roughly proportional to VoI. This concept might be used when defining a heuristic policy, which behaves similarly to the effective communication policy but is much simpler to design and implement. Note that this condition includes two separate cases in which a level C observer chooses not to transmit, while level A and B transmitters would send precise data:

- The action corresponding to the prior belief is the same as the one after the updating message. In this case, the VoI of the communicated message is low and thus we can lower ℓ_t;
- The action is different after the communicated message, but the long-term rewards are close enough that the robot is not going to benefit too much from choosing the other action.



(a) Age of Information (AoI)=0. (b) AoI=1. (c) AoI=2. (d) AoI=3. (e) AoI=4.

Figure 3.14: Level A observer action distribution for different robot action entropy levels with $\beta = 1$.

> Even in this case, sending less information is not going to affect the control performance significantly.

These cases cannot be taken into account in level A and B. Indeed, the level A policy shown in Fig. 3.11 (c) tends to allocate communication resources in the states where the picture is changing more rapidly, so that the memory available to the robot is less useful to estimate the current observation, regardless of the correct action. As the cart speed $\dot{x}$ increases along the y-axis, the number of bits increases too. The same reasoning can be applied to the results in Fig. 3.12.

Another general principle that we can deduce for an effective policy is that it should be aware of variations of the value function with respect to the belief. If the value function is strongly affected by small perturbations of the belief, then the effective policy should communicate more information in order to reduce the discrepancy between $\xi_t^{(o)}$ and $\xi_t^{(r)}$. This reasoning can be intuitively understood by looking at the differential of the robot's value function $Q\left(\xi_t^{(r)}, \pi^{(r)}\left(\xi_t^{(r)}\right)\right)$ with respect to changes in its belief distribution $\xi_t^{(r)}$. When this value is big, an inaccurate estimation of the state would cause a poor estimation of the value function, which may in turn cause the robot to choose a low-quality action.

In Fig. 3.13, we provide an analysis of the communication strategy with respect to different AoI values. This allows to show how the memory of the robot and of the observer plays a crucial role on the communication decisions. In particular, we consider five values of the AoI: AoI $= 0$ indicates that a message of any length was transmitted in the previous time step. AoI $= N$ with $N \in \{1, 2, 3, 4\}$ means that no messages have been received by the observer for N time steps since the last received message. This is a measure of how up to date the memory of the robot is, allowing us to evaluate the next choice of the observer for a given age. We then consider the distribution of the observer actions (y-axis) with respect to different ranges of the robot actions entropy (x-axis). This means that, for each entropy interval, we count the number of times each action is performed, in order to obtain an empirical distribution. The columns are normalized so that each cell shows the probability that the observer chooses a specific ℓ_t whenever the robot action entropy falls within the corresponding interval, for different values of the AoI.

Fig. 3.13 clearly shows that, if there was a transmission in the previous time step (AoI $= 0$), it

is very unlikely that the system is going to be updated again in the current time step. Conversely, the observer chooses to communicate if the AoI = 1, with an exception if the system is in a very low entropy state, in which case the probability of communicating using Φ_4 is similar to the one corresponding to action $\Phi_\varnothing$. If we look at the behavior for higher values of the AoI, we can notice a general trend: communication is more likely to happen in higher entropy states than in lower entropy ones. This shows that the observer policy understands the cases where the state has to be precisely estimated by the robot to choose its action correctly. Fig. 3.14 shows that the level A policy allocates communication resources without considering the entropy of the control actions. The value of β for this was chosen to get a similar overall bitrate (and, as we discussed, a similar overall action distribution) to the Level C case with $\beta = 0.15$. However, we can see that the trend holds for different values of β by looking at the Fig. 3.13: if we decrease the value of β, the observer tends to transmit more often, and use higher message lengths when it transmits, but the general tendency to transmit more whenever the robot action entropy is high clearly holds. This final analysis allows us to get an easy heuristic for effective communication when the value function is not available or cannot be learned.

3.3.5 Conclusion

In this work, we presented a dynamic feature compression scheme that can exploit an ensemble VQ-VAE to solve the semantic and effective communication problems. The dynamic scheme outperforms fixed quantization, and can be trained automatically with limited feedback, unlike emergent communication models that are unable to deal with complex tasks. The choices made by the observer are clearly tied to the control policy of the robot it aims to help, significantly outperforming a simpler optimization that does not take into account the semantic and effective problems. We also analyzed the optimal policies to draw insights on their decisions, showing that the Level C optimization indeed considers the robot's policy.

A natural extension of this model is to consider more complex tasks and wider communication channels, corresponding to realistic control scenarios, or scenarios with multiple transmitters with partial information about each other and the robot. Another interesting direction for future work is to consider joint training of the robot and the observer, or cases with partial information available at both transmitter and receiver.

3.4 Supplementary and Proofs

3.4.1 Proofs

Regret Analysis - Theorem 20

In this section we prove Theorem 20. As explained Sec. 3.2.3, we align the interactions between agents and environment in time, and consider virtual rounds $t \in \{1, \dots, J \cdot N\} = \mathcal{T}$, and rewrite the Bayesian system regret as follows

$$\begin{aligned}
\mathrm{BR}(\pi^{TS}, J) &= \mathbb{E}\left[\sum_{n\in\mathcal{N}}\sum_{j=1}^{J} \mu(s_{j,n}, a^*(s_{j,n}^n)) - \mu\left(s_{j,n}, A_{j,n}\right)\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{NJ} \mu(s_t, a^*(s_t)) - \mu\left(s_t, A_t\right)\right] \\
&= \mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{t_s\in\mathcal{T}_s} \mu(s_{t_s}, a^*(s_{t_s})) - \mu\left(s_{t_s}, A_{t_s}\right)\right]
\end{aligned}$$

where $\mathcal{T}_s = \{t \in \mathcal{T} : s_t = s\}$, i.e., all time-steps where context s was sampled, and $|\mathcal{T}_s| = T_s$. We now define the upper and lower bounds for the reward average $\mu(s, a)$, which hold with high probability, and are used to bound the average per-round regret

$$U_t(s,a) = \hat{\mu}_{t-1}(s,a) + \sqrt{\frac{2 + 6\log T_s}{\phi_{t-1}(s,a)}} \tag{3.25}$$

$$L_t(s,a) = \hat{\mu}_{t-1}(s,a) - \sqrt{\frac{2 + 6\log T_s}{\phi_{t-1}(s,a))}} \tag{3.26}$$

where $\hat{\mu}_t(s,a)$ is the empirical average return of a with context s at time t, and $\phi_t(s,a)$ is related to the number of times arm a has been pulled until time t with context s, and will be better explained later. We now use *Proposition 1* from [214], which allows us to write

$$\mathrm{BR}(\pi^{TS}, T) = \mathbb{E}\left[\sum_{t\in\mathcal{T}} U_t\left(s_t, A_t\right) - \mu\left(s_t, A_t\right)\right] + \mathbb{E}\left[\sum_{t\in\mathcal{T}} \mu\left(s_t, a^*(s_t)\right) - U_t\left(s_t, a^*(s_t)\right)\right], \tag{3.27}$$

and apply *Proposition 2* in [214] obtaining, whenever $T > SKN$,

$$\mathrm{BR}(\pi^{TS}, T) \leq \sum_{s\in\mathcal{S}}\sum_{t_s\in\mathcal{T}_s} \mathbb{E}\left[U_{t_s}\left(s, A_{t_s}\right) - L_{t_s}\left(s, A_{t_s}\right)\right] + SKN. \tag{3.28}$$

We now consider the single context regret. For every $s \in \mathcal{S}$, we define $\mathcal{T}_s^a = \{t_s \in \mathcal{T}_s : A_{t_s} = a\}$ and $|\mathcal{T}_s^a| = T_s^a$, bounding the quantity

$$\begin{aligned}\sum_{t_s \in \mathcal{T}_s} U_{t_s}(s, A_{t_s}) - L_{t_s}(s, A_{t_s}) &= \sum_{a \in \mathcal{A}} \sum_{t_s^a \in \mathcal{T}_s^a} U_{t_s^a}(s,a) - L_{t_s^a}(s,a) \\ &\overset{(a)}{\leq} \sum_{a \in \mathcal{A}} \left[1 + 2\sqrt{2 + 6 \log T_s} \sum_{t_s^a \in \mathcal{T}_s^a} (1 + \phi_{t_s^a - 1}(s,a))^{-\frac{1}{2}} \right]\end{aligned}$$

Here (a) is a consequence of the bounds defined in Eq. (3.25). In standard multi-armed bandit (MAB) analysis, as in [214], $\phi_t(s,a)$ is basically the number of times arm a has been sampled up to time t with context s, that we denote with $C_t(s,a)$. This quantity is critical to bound the intervals $U_t(s,a) - L_t(s,a)$. However, in our formulation, we define $\phi_t(s,a) = C_{j(t)}(s,a)$, where

$$j(t) = \lfloor t/N \rfloor \cdot N$$

is the last time the policy has been updated, i.e., at the end of the previous round. Now, $\phi_t(s,a) \geq C_t(s,a) - N$, since it is not possible to sample more than N times the same arm in one round, given that the number of agents is N. However, the quantity $(1 + C_t(s,a) - N)^{-\frac{1}{2}}$ is not defined for $C_t(s,a) < N$. If $C_t(s,a) < N$, we can always write $\phi_t(s,a) \geq \frac{C_t(s,a)}{N} + 1$.

Finite-Time Analysis With the observations above, we can now prove an upper bound for the finite-time regret. We first notice that

$$\begin{aligned}\sum_{t_s^a \in \mathcal{T}_s^a} (1 + \phi_{t_s^a - 1}(s,a))^{-\frac{1}{2}} &\leq \sum_{t_s^a \in \mathcal{T}_s^a} \left(2 + \frac{C_{t_s^a - 1}(s,a)}{N} \right)^{-\frac{1}{2}} \\ &= \sum_{j=0}^{T_s^a - 1} \left(2 + \frac{j}{N} \right)^{-\frac{1}{2}} \\ &\leq \sum_{j=0}^{T_s^a - 1} \left(\frac{j+1}{N} \right)^{-\frac{1}{2}} \\ &= \sum_{j=1}^{T_s^a} \left(\frac{j}{N} \right)^{-\frac{1}{2}}.\end{aligned}$$

We can now use the integral bound to find

$$\sum_{j=1}^{T_s^a} \left(\frac{j}{N} \right)^{-\frac{1}{2}} \leq \sqrt{N} \int_{\tau=0}^{T_s^a} \tau^{-\frac{1}{2}} d\tau = 2\sqrt{N}\sqrt{T_s^a}$$

If we put all together, we obtain

$$\begin{aligned}
\mathrm{BR}(\pi, T) &\leq \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} 1 + 4\sqrt{2+6\log T_s}\sqrt{N}\sqrt{T_s^a} \\
&\leq \sum_{s\in\mathcal{S}} K + 4\sqrt{2+6\log T_s}\sqrt{N}\sum_{a\in\mathcal{A}}\sqrt{T_s^a} \\
&\overset{(a)}{\leq} \sum_{s\in\mathcal{S}} K + 4\sqrt{(2+6\log T_s)NK\sum_{a\in\mathcal{A}} T_s^a} \\
&= KS + \sum_{s\in\mathcal{S}} 4\sqrt{(2+6\log T_s)KNT_s} \\
&\leq KS + 4\sqrt{(2+6\log T)KN}\sum_{s\in\mathcal{S}}\sqrt{T_s} \\
&\overset{(b)}{\leq} 2KS + 4\sqrt{(2+6\log T)KNST}.
\end{aligned}$$

The equalities (a) and (b) come from the Cauchy-Shwartz inequality, and the definitions of T_s^a, T_s, and T.

Asymptotic Regret Analysis However, in the asymptotic case $N \to \infty$, we get rid of the first constant terms when arms are pulled less than N times. Consequently, we can use the bound $\phi_t(s,a) \geq C_t(s,a) - N$, and follow the same analysis for the finite-time case. We can see that the factor T does not scale with N, obtaining

$$\mathrm{BR}(\pi, T) \in \mathcal{O}\left(\sqrt{KTS\log T}\right).$$

Achievable Rate - Lemma 21 and Lemma 22

In this section we provide the detailed proofs of Lemma 21 and Lemma 22. To this end, we denote by $H_q(X)$ and $I_q(X;Y)$ the entropy and the mutual information computed with respect to the probability distribution q. We recall that $H(A^*)$ is the entropy of the optimal actions, i.e., computed with respect to $\pi^*(a) = \sum_{s\in\mathcal{S}} P(s)\pi(a^*|s)$, where $\pi^*(a|s)$ is the optimal policy in context s.

We start by proving the following lemma.

Lemma 25. *Assuming that Thompson Sampling policy $\pi_j(a|s)$ achieves sub-linear Bayesian system regret, then $\lim_{j\to\infty} I_{\pi_j}(S;A) = \lim_{j\to\infty} H_{\pi_j}(A) = H(A^*)$.*

Proof. First of all we notice that, in order to achieve sub-linear Bayesian system regret, it is necessary to achieve sub-linear regret in all contexts $s \in \mathcal{S}$, given the assumption that $\forall s \in \mathcal{S}\ P_S(s) > 0$. We then write $I_{\pi_j}(S;A) = H_{\pi_j}(A) - H_{\pi_j}(A|S)$. Following Theorem 2 from [133], if $\pi_j(a|s)$

achieves sub-linear regret, then $\forall s \in \mathcal{S}$

$$\lim_{j\to\infty} \pi_j(a = a^*|s) = 1 \quad \text{when } a^* \text{ is the optimal arm,}$$
$$\lim_{j\to\infty} \pi_j(a = a'|s) = 0 \quad \text{when } a' \neq a^*.$$

Consequently, in the limit, $\pi_j(a|s)$ is a deterministic function, thus

$$\lim_{t\to\infty} H_{\pi_t}(A^*|S) = 0,$$

which concludes our proof. □

We start by proving Lemma 22, which we repeat below.

Lemma 22 If $R > H(A^*)$, then it is possible to achieve sub-linear Bayesian system regret, in the limit $N \to \infty$.

Proof. We denote with R_{π_j} the rate needed to perfectly convey the Thompson Sampling (TS) policy to the controller at round j, and let $\epsilon > 0$ s.t. $R = H(A^*) + \epsilon$, where R is the available communication rate. We now provide a scheme that guarantees sub-linear regret.

If the policy π_j generated from TS has $R_{\pi_j} \leq H(A^*)\ \forall j = 1, \ldots, J$, then Theorem 20 ensures sub-linear Bayesian system regret if sampling with π_j. If $\exists j$ such that $R_{\pi_j} > R$, generate parameters ρ_j such that $\rho_j \in o(1)$ and $\sum_{j=1}^{\infty} \rho_j = \infty$, as explained in [202], Theorem 3. Then, with probability ρ_t play a uniformly at random, and with probability $1 - \rho_t$, play according to a policy $Q_j(a|s)$, which satisfies the rate-distortion constraint $I_{Q(a|s)}(S; A) \leq R$, which can be transmitted to the controller, using as distortion measure the reverse KL divergence $d(Q_{SA}, \pi_{SA}) = \mathbb{E}_{P_S}\left[D_{KL}(Q(\cdot||S)||\pi(\cdot||S))\right]$. Following Lemma 14 in [213], with this strategy enough exploration is guaranteed for the posterior policy, i.e., the one stored by the decision-maker, to concentrate. Consequently, by Lemma 25, there exists a finite j_0 s.t. $\forall j > j_0$, $R_{\pi_j} < H(A^*) + \epsilon$, in the limit $N \to \infty$. This means that, for the first j_0 rounds, both the target and the approximating policies would play sub-optimal arms with non-zero probabilities. Then, $\forall j > j_0$, it is possible to play the exact TS policy, leading to the optimal decisions for all future steps, and hence, to a sub-linear regret. The above procedure holds $\forall \epsilon > 0$, and so $\forall R > H(A^*)$. □

As we can see, the strategy that achieves sub-linear regret consists in using TS at the decision-maker, which updates the posteriors according to the Bayes rule, and from which a target policy π^* is computed. Then, the decision-maker computes an approximate policy $Q_j(a|s)$ as indicated in the proof, and it samples the arms according to $Q_j(a|s)$. We further notice that the limit $H(A^*)$ serves as a lower bound for practical schemes, as to achieve it we need $N \to \infty$.

We observe that Theorem 2 in [133] states that, if the TS strategy achieves sub-linear regret, then the policy converges to the deterministic policy selecting with probability one the optimal action. However, the converse is not always true in general, i.e., there could exist policies that play sub-optimal arms infinitely many times as $j \to \infty$, and still achieve sub-linear regret. The point is that such policies must play the optimal arms for most of the rounds, and could pull

sub-optimal arms for a sub-linear amount of rounds. However, to play optimally in one round, the decision-maker needs $R \geq R_{\pi^*} = H(A^*)$. We are now ready to prove Lemma 21.

Lemma 26. *If $R < H(A^*)$, it is not possible to achieve sub-linear Bayesian system regret.*

Proof. As explained above, to play optimally in one round, the decision-maker needs $R \geq R_{\pi^*} = H(A^*)$, in the limit $N \to \infty$. If $R < H(A^*)$, as a consequence of Eq. (3.6), the policy $Q(a|s)$ conveyed to the controller has non-zero distortion $d(Q_{SA}, \pi^*_{SA}) = D > 0$. If we take the L_1 norm as distortion measure, $Q(a|s)$ would sample a sub-optimal arm with constant probability of at least D in every round $j = 1, \ldots, J$. Consequently, sub-linear regret cannot be achieved. □

Policy Compression Schemes

To compute the optimal policy that solves Eq. (3.6) with a specific distortion function, we applied the well known Blahut-Arimoto iterative algorithm [69] that, given the considered distortion functions, is guaranteed to converge to the solution [70].

We rewrite the optimization objective of Eq. (3.6) as a double minimization problem (Sec. 10.8, [69])

$$R(D) = \min_{\tilde{Q}(a)} \min_{Q_{A|S}: d(Q_{SA}, P_{SA}) \leq D} \sum_{s,a} P_S(s) Q(a|s) \log_2 \frac{Q(a|s)}{\tilde{Q}(a)}. \tag{3.29}$$

Following (Lemma 10.8.1, [69]), the marginal $\tilde{Q}(y) = \sum_x P(x)Q(y|x)$ has the property

$$\tilde{Q}(y) = \arg\min_{Q(y)} D_{KL}(P(x)Q(y|x)||P(x)Q(y)), \tag{3.30}$$

that is, it minimizes the KL-divergence between the joint and the product $P(x)Q(y)$. This means that $\tilde{Q}(a)$ obtained by solving Eq. (3.29) is indeed the marginal over the arms induced by $Q(a|s)$. Exploiting this formulation, it is possible to apply the iterative Blahut-Arimoto algorithm to solve the problem and find the solution [69]. The process is initialized by setting a random $\tilde{Q}_0(a)$, which is used as a fixed point to compute

$$Q_1^*(a|s) = \mathrm{argmin}_{Q_{A|S}: d(Q_{SA}, P_{SA}) \leq D} \sum_s P(s) \sum_a Q(a|s) \log_2 \frac{Q(a|s)}{\tilde{Q}_0(a)}. \tag{3.31}$$

From $Q_1^*(a|s)$, we compute the optimal $\tilde{Q}_1(a)$ by solving Eq. (3.30), which is simply the marginal $\tilde{Q}_1(a) = \sum_s P(s) Q_1^*(a|s)$. The process is iterated until convergence. We now solve the inner minimization problem, i.e., Eq. (3.31) with fixed $\tilde{Q}(a)$ and distortion $\mathbb{E}_{P_S}[D_\alpha(Q, \pi)]$, for $\alpha \to 1$, and $\alpha \to 0$.

Reverse KL Divergence ($\alpha \to 0$) To solve this problem, we solve the related Lagrangian

$$\mathcal{L}(Q(a|s), \lambda, \mu) = \sum_s P(s) \sum_a Q(a|s) \log \frac{Q(a|s)}{\tilde{Q}_a} + \lambda \left(\sum_s P(s) \sum_a Q(a|s) \log \frac{Q(a|s)}{\pi(a|s)} - D \right) + \\ + \mu \left(\sum_s P(s) \sum_a Q(a|s) - 1 \right)$$

where the Lagrangian multiplier λ has to be optimized to meet the constraints on the divergence, whereas μ ensures that the solution is a probability distribution, i.e., the elements sum to one. The positivity constraints on the terms are already satisfied by the fact that the solution has an exponential shape. We first take the derivative of the Lagrangian w.r.t. to the terms $Q(a|s)$ and set it to zero

$$\frac{\partial \mathcal{L}(Q(a|s), \lambda, \mu)}{\partial Q(a|s)} = P(s) \log \frac{Q(a|s)}{\tilde{Q}(a)} + P(s) - \sum_{s'} P(s') Q(a|s') \frac{P(s)}{\tilde{Q}(a)} + \\ + \lambda P(s) \left(\log \frac{Q(a|s)}{\pi(a|s)} + 1 \right) + \mu P(s) = 0$$

finding

$$\begin{aligned} \log \frac{Q(a|s)^{1+\lambda}}{\tilde{Q}(a) \pi(a|s)^{\lambda}} &= -(\lambda + \mu) \\ Q(a|s)^{1+\lambda} &= e^{-(\mu+\lambda)} \tilde{Q}(a) \pi(a|s)^{\lambda} \\ Q(a|s) &= e^{\frac{-(\mu+\lambda)}{1+\lambda}} \tilde{Q}(a)^{\frac{1}{1+\lambda}} \pi(a|s)^{\frac{\lambda}{1+\lambda}}. \end{aligned}$$

We now define $\gamma := \frac{1}{1+\lambda}$, $\gamma \in [0, 1]$, and obtain the distribution

$$Q_\gamma(a|s) = \frac{\tilde{Q}(a)^\gamma \pi(a|s)^{1-\gamma}}{\sum_{a' \in \mathcal{A}} \tilde{Q}(a')^\gamma \pi(a'|s)^{1-\gamma}}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{3.32}$$

By the convexity of KL-Divergence and its triangular inequality, we know the solution lies on the boundary of the constraints, i.e., when $\mathbb{E}_{P_S} [D_{KL}(Q_\gamma || \pi)] = \delta$.

Forward KL Divergence ($\alpha \to 1$) The derivative of the Lagrangian is (here the normalization factor is added in the end)

$$\frac{\partial \mathcal{L}(Q(a|s), \lambda)}{\partial Q(a|s)} = P(s) \log \frac{Q(a|s)}{\tilde{Q}(a)} - \lambda P(s) \frac{\pi(a|s)}{Q(a|s)}$$

and setting it to zero leads to

$$\begin{aligned}\frac{\partial \mathcal{L}\left(Q(a|s), \lambda\right)}{\partial Q(a|s)} &= 0 \\ \log \frac{Q(a|s)}{\tilde{Q}(a)} - \lambda \frac{\pi(a|s)}{Q(a|s)} &= 0 \\ \log \frac{\tilde{Q}(a)}{Q(a|s)} + \lambda \frac{\pi(a|s)}{Q(a|s)} &= 0.\end{aligned}$$

We now we define $x := \frac{1}{Q(a|s)},\ \alpha := \lambda\pi(a|s),\ \beta := 1$, and $\gamma := \log \tilde{Q}(a)$, obtaining

$$\begin{aligned}\alpha x + \beta \log x + \gamma &= 0 \\ \alpha x + \log \alpha x + \gamma - \log \alpha &= 0 \\ e^{\alpha x} \alpha x &= \alpha e^{-\gamma} \\ x &= \frac{1}{\alpha} \mathbf{W}_0\left(\alpha e^{-\gamma}\right)\end{aligned}$$

where $\mathbf{W}_0(\cdot)$ is the Lambert function [147]. We can now replace the introduced variables with the original terms and normalize, obtaining

$$Q_\lambda(a|s) = \frac{\lambda\pi(a|s)\mathbf{W}_0\left(\frac{\lambda\pi(a|s)}{\tilde{Q}(a)}\right)}{\sum_{a' \in \mathcal{A}} \lambda\pi(a'|s)\mathbf{W}_0\left(\frac{\lambda\pi(a'|s)}{\tilde{Q}(a')}\right)},$$

with λ such that $\mathbb{E}_{P_S}\left[D_{KL}(\pi||Q_\lambda)\right] = \delta$.

Clustering Compression Schemes

Reverse KL Divergence Again, we compute the optimal centroids by solving the Lagrangian

$$\mathcal{L}(\mu_a^c, \lambda) = \sum_{s \in \mathcal{S}_c} P(s) \sum_{a \in \mathcal{A}} \mu_a^c \log \frac{\mu_a^c}{\pi(a|s)} + \lambda \left(\sum_{a \in \mathcal{A}} \mu_a^c - 1 \right)$$

taking its derivative and solving the equality

$$\frac{\partial \mathcal{L}(\mu_a^c, \lambda)}{\partial \mu_a^c} = \sum_{s \in \mathcal{S}_c} P(s) \left(\log \frac{\mu_a^c}{\pi(a|s)} + 1 \right) + \lambda = 0$$

finding

$$\log \mu_a^c A(\mathcal{S}_c) = \sum_{s \in \mathcal{S}_c} P(s) \log \pi(a|s) + A(\mathcal{S}_c) + \lambda$$
$$\log \mu_a^c = \sum_{s \in \mathcal{S}_c} \frac{P(s)}{A(\mathcal{S}_c)} \log \pi(a|s) + 1 + \frac{\lambda}{A(\mathcal{S}_c)}$$
$$\mu^c = \frac{\prod_{s \in \mathcal{S}_c} \pi(a|s)^{\frac{P(s)}{A(\mathcal{S}_c)}}}{Z},$$

where Z is the normalizing factor, obtaining the shape expressed in Eq. (3.15).

Forward KL Divergence In this case, the Lagrangian is

$$\mathcal{L}(\mu_a^c, \lambda) = \sum_{s \in \mathcal{S}_c} P(s) \sum_{a \in \mathcal{A}} \pi(a|s) \log \frac{\pi(a|s)}{\mu_a^c} + \lambda \left(\sum_{a \in \mathcal{A}} \mu_a^c - 1 \right).$$

We take the derivative, and set it equal to zero

$$\frac{\partial \mathcal{L}(\mu_a^c, \lambda)}{\partial \mu_a^c} = \sum_{s \in \mathcal{S}_c} P(s) \left(-\frac{\pi(a|s)}{\mu_a^c} \right) + \lambda = 0$$

finding

$$\mu_a^c = \frac{\sum_{s \in \mathcal{S}_c} P_S(s) \pi(a|s)}{Z}$$

where Z is the normalizing factor.

3.4.2 Supplementary: The Information Bottleneck View

We can also consider another perspective on the observer's choices, using information bottleneck theory. We define a *sufficient statistic* $i(s)$ of any given state $s \in \mathcal{S}$, which is enough to determine the robot's performance in that state. Denoting the number of bits required to represent a realization of random variable X as $b(X)$, we consider a case in which:

$$b(i(S)) < b(S) < b(O).$$

Indeed, the observation may contain much more information than needed to estimate the state [69], and lossily compressing the message to preserve the relevant information, removing redundant or irrelevant details, can ease communication requirements without losing performance. We can also observe that $i(S) \to S \to O$ is a Markov chain. The random quantity $i(S)$ represents the minimal description of the system with respect to the robot's task, i.e., no additional data computed from S adds meaningful information for the robot's policy. The state S may also include task-irrelevant physical information on the system. However, both S and $i(S)$ are unknown quantities, as the

observer only receives a noisy and high-dimensional representation of S through O. This is a well-known issue in DRL: in the original paper presenting the Deep Q-Network (DQN) architecture [175], the agent could only observe the screen while playing classic arcade videogames, and did not have access to the much more compact and precise internal state representation of the game. Introducing communication and dynamic encoding adds another layer of complexity.

We can then consider the case in which communication is limited to a maximum length of L bits, i.e., to $2^{L+1} - 1$ messages, considering all possible lengths lower than or equal to L, including no communication. The channel is ideal, i.e., instantaneous and error-free, but it includes a constant cost per bit β, as in the observer reward we gave in the previous section. Consequently, the problem introduces an information bottleneck between the observation O_t and the estimate $\hat{o}_t$ that the robot can make, based on the message M_t conveyed through the channel. If we define a distortion measure over the observation space $d_A : \mathcal{O}^2 \to \mathbb{R}^+$, any communication introduces a non-zero distortion $d_A(o, \hat{o})$ whenever $b(o) > L$, whose theoretical asymptotic limits are given by rate-distortion theory [69]. If we also consider *memory*, i.e., the use of past messages in the estimation of $\hat{o}$, the mutual information between o and the previous messages can be used to reduce the distortion, improving the quality of the estimate.

In the semantic problem, the aim is to extrapolate the real physical state of the system S_t from the compressed observation M_t, which can be a complex stochastic function. In general, the real state lies in a low-dimensional semantic space $\mathcal{S}$. The term semantic is motivated by fact that, in this case, the observer is not just transmitting pure sensory data, but some meaningful piece of physical information about the system. Consequently, the distortion to be considered in this case can be represented by a measure $d_B : \mathcal{S}^2 \to \mathbb{R}$ over the semantic space, so that the distortion $d_B(\hat{s}_t^{(o)}, \hat{s}_t^{(r)})$ is computed between the observer's best estimate of the state and the one performed by the robot based on M_t, and on its memory of past messages. Finally, to be even more efficient and specific with respect to the task, the observer may optimize the message M_t to minimize a distortion measure $d_C \left(i\left(\xi_t^{(o)}\right), i\left(\xi_t^{(r)}\right)\right)$ between the effective representation of the observer's belief on the state, which contains only the task-specific information, and the knowledge available to the robot. Naturally, any message instance $m_t \in \mathcal{M}$ must be at most L bits long, in order to respect the constraint. However, defining the sufficient statistic $i\left(\xi_t^{(o)}\right)$ may be highly complex and problem-dependent, and using the robot's reward as a direct performance measure is significantly more direct, with the same guarantees.

4

Applications of Distributed Learning

If in the first two chapters the main focus was to analyze and optimize distributed learning algorithms, in this final part of the thesis some applications of distributed learning are studied.

In Section 4.1 and Section 4.2 the problem of distributed resource allocation in wireless networks is analyzed. In particular, the goal is to provide Ultra-Reliable and Low-Latency Communications (URLLC) in Industrial Internet of Things (IIoT) networks, for which standard 5G protocol are not sufficient. The problem is formulated using the multi-agent contextual multi-armed bandit (CMAB) framework, for which a novel algorithm is introduced in Section 4.2. By adopting a distributed and data-driven approach, we demonstrate that better communication latency performance can be obtained, with respect to those of actual standards implemented by 5G protocols, suggesting the adoption of machine learning (ML) techniques in the design of 6G networks.

In the end, we explore the application of multi-agent reinforcement learning (MARL) to control a swarm of drones surveiling a 2-D area with some sparse hot targets to be monitored. In this case the task is to learn a policy to control the trajectories of the drones considering noisy communications among them, as well as obstacles in the map. We show through empirical simulations that it is indeed possible to accomplish the task of reaching the targets faster than competitive baselines.

4.1 Distributed Resource Allocation for URLLC in IIoT Scenarios: A Multi-Armed Bandit Approach

4.1.1 Introduction

With early 5th generation (5G) deployments already rolled out, the research community is discussing use cases, requirements, and enabling technologies towards sixth generation (6G) sys-

tems [96]. Among other services, 6G will introduce new communication interfaces and innovative architectures to support the Industrial Internet of Things (IIoT) in 2030 and beyond, where the 6G network connects sensors and machines in factories, plants, mines, to enable analytics, diagnostics, monitoring, asset tracking, as well as process, regulatory, supervisory, and safety control [151]. In this context, the need for robots to complete cooperative operations that require high precision and coordination in real time comes with its own set of requirements, e.g., in terms of reliability (up to 99.99999%) and latency (below 1 ms, or even 0.1 ms, in the radio part), making it crucial to support Ultra-Reliable and Low-Latency Communications (URLLC) in the industrial domain [263]. The factory of the future will further operate to support high-density deployments of machines and end users.

In this context, the time introduced by the Radio Access Network (RAN) operations, from routing and scheduling to resource allocation and modulation, represents one of the most impactful latency components. Specifically, a *centralized pre-configured* scheduling protocol usually requires the prior exchange of scheduling requests (grants) to (from) the Next Generation Node B (gNB), which is not compatible with URLLC in IIoT scenarios [74, 269]. To partially address this issue, 3GPP NR supports *semi-persistent* and *grant-free* communication in the uplink (UL) [5], in which the network pre-allocates radio resources, thereby eliminating the need for User Equipments (UEs) to wait for network grants before transmission. However, reserving resources to dedicated UEs can be inefficient if traffic demands are aperiodic [169], and it is not possible to anticipate when resources will be needed [48].

Another solution is to design a *user-centric* architecture (as foreseen in 6G [257]) in which end machines make autonomous decisions, "disaggregated" from the network [186]. Along these lines, in this work we explore the feasibility of a decentralized/distributed scheduling algorithm that, exploiting machine learning (ML) technologies, allows UEs to optimize their UL transmission strategies by autonomously selecting the available physical resources. This framework is able to learn from the application, and could work well even considering architectures for IIoT scenarios in which communication is on the sidelink, with no or limited support from the gNB [12].

Despite this potential, however, distributed scheduling may create collisions during communication, raising the question of whether this approach is compatible with URLLC applications. To this aim, we apply the multi-armed bandit (MAB) theory [226] to evaluate how autonomous machines should select transmission resources based on previous scheduling decisions and the effect they produced on the network in terms of reliability. While the MAB approach is well known, most related work focused on Deep Learning (DL) [160], cellular [102], or IoT [120] networks. In turn, we consider a UL scenario modeled according to the "Motion Control" 5G-ACIA geometry (in which a remote server sends commands to control the moving parts of machines), thus ensuring that our results are representative of a typical IIoT environment. Other notable papers consider vehicular scenarios [268, 156], where the target is to enable URLLC for vehicle-to-vehicle communications via Deep Reinforcement Learning (DRL). However, we argue that for IIoT use cases, state-of-the-art MAB algorithms may better exploit the strong correlation typical of the industrial environment while, at the same time, reducing the computational complexity and training time to converge to optimal solutions, compared to more sophisticated DRL alternatives.

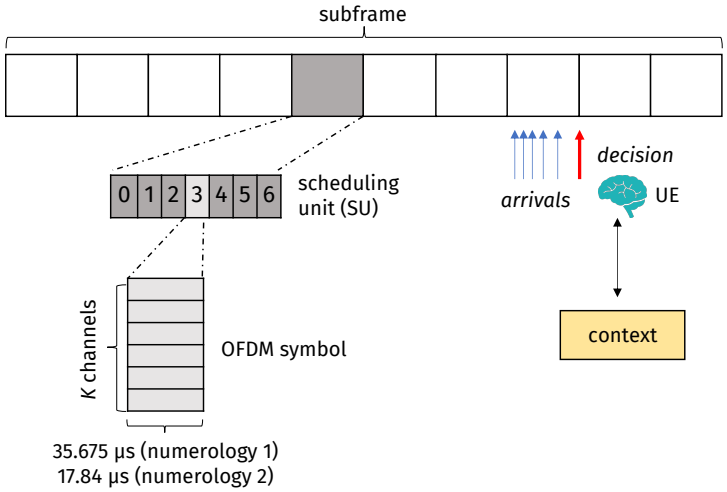


Figure 4.1: Transmission structure.

We perform simulations with both periodic and aperiodic traffic, and as a function of the UEs' density and spatial distribution, the traffic periodicity (thereby modeling aggressive or conservative applications), and the transmit power, thus considering a low-power performance regime. From our results, we conclude that the Thompson Sampling agent [214] is a promising candidate method to minimize the collision probability even in the presence of unscheduled transmissions.

4.1.2 Problem Formulation and System Model

We consider an Orthogonal Frequency Division Multiplexing (OFDM) system in which devices, also denoted as agents in machine learning parlance, are located in a factory environment, and have to autonomously choose the orthogonal channel to be used for UL transmissions. The time domain is discretized into intervals of duration equal to the OFDM symbol (with a Scheduling Unit (SU) consisting of 7 OFDM symbols), whose duration depends on the adopted NR numerology. The frequency domain is also discretized into K orthogonal channels, whose size depends on the available bandwidth B and the subcarrier spacing Δf.

At the beginning of each SU, the agents make their scheduling decisions, that is the channel to be used for transmission, as shown in Fig. 4.1. Unlike in a centralized pre-configured resource allocation approach, in which radio resources are scheduled by the gNB via scheduling grants, we study the feasibility of a decentralized algorithm based on ML in which each agent autonomously optimizes its channel selection policy relying only on the gNB feedback, without prior ad hoc message exchange with the gNB itself. The rationale behind this scheme is to exploit the underlying correlations typical of the IIoT traffic to avoid the transmission of centralized scheduling grants, thus reducing the end-to-end latency and promoting URLLC.

If multiple agents use the same physical channel during a specific SU, we assume that their packets are lost due to a *collision* event. At the end of each SU, the gNB broadcasts a mes-

sage indicating in which channel(s) data were successfully received. This message is used by the pool of agents to optimize their subsequent decision strategies, and achieve coordination without communication.

We formalize the problem using the MAB framework, which is used to model many sequential decision processes in computer science and engineering [226]. In this particular multi-agent scenario, there are N agents, i.e., the N UEs, interacting with the same environment. Whenever an agent $n \in \{1, \ldots, N\}$ generates a new packet during SU t, it schedules its transmission at the beginning of SU $t+1$, choosing one among the K available channels, which will be used for transmission for the whole SU duration. According to the MAB notation, we refer to the action of using channel $k \in \mathcal{K} = \{1, \ldots, K\}$ as "playing the arm" k. At the end of SU $t+1$, the message received from the gNB is converted into a reward $r_{n,t}$, indicating whether or not the transmission was successful, i.e., $r_{n,t} = 1$ or $r_{n,t} = 0$, respectively: maximizing the reward implies transmitting the data successfully in low latency, as there is no need to exchange scheduling grants between the UEs and the gNB, leading to the URLLC objective. In our model, we assume that the reward behind each action is sampled from a Bernoulli distribution with unknown parameter $\mu_n(k_{n,t})$, which depends on the action taken by the agent, and captures the probability of the other agents transmitting at the same time. Thus, in each SU t, agent n samples an action $k \in \mathcal{K}$ according to its policy $\pi_n : \mathcal{H}^{t-1} \rightarrow \Delta_K$, which is, in general, a map from history $H_n(t-1) = \{(k_{1,n}, r_{1,n}), \ldots, (k_{t-1,n}, r_{t-1,n})\} \in \mathcal{H}^{t-1}$ to a probability distribution over the action set $\mathcal{K}$, where Δ_K denotes the K-simplex. The history vector $H_n(t-1)$ is used by the agent to optimize its policy π_n, so as to maximize the expected cumulative reward $R(\pi_n, T) = \mathbb{E}_{\pi_n}\left[\sum_{t=1}^{T} \mu_n\left(k_{n,t}\right)\right]$.

4.1.3 multi-armed bandit (MAB) Agents

To solve the problem in Section 4.1.2 and maximize the reward, many algorithms have been proposed in the literature over the past years [226]. In this work, we study the performance of different MAB agents to solve the problem of distributed resource allocation, in the specific context of URLLC for IIoT.

Random Agent (RA) It implements the simplest decision scheme, and is used as a lower bound. Nonetheless, it represents well the case of 5G NR grant-free scheduling, where the access decision is random, and re-transmissions are optimized to achieve reliability [185]. In particular, in each SU, the RA selects uniformly, at random, one of the K arms, and no learning is involved.

UCB Agent (UCB-A) It implements the upper confidence bound (UCB) algorithm [148], i.e., the agent plays, in each SU t, the arm k_t such that

$$k_t = \text{argmax}_{k \in \mathcal{K}} \left[Q_t(k) + c\sqrt{\frac{\log t}{n_t(k)}} \right], \tag{4.1}$$

where $Q_t(k)$ is the empirical average at step t of the experienced rewards for arm k, $n_t(k)$ is the number of times arm k has been played until time step t, and c is an exploration parameter to be optimized. In Eq. (4.1), $Q_t(k)$ represents the *exploitation* part, as it is related to the past experience, while $\sqrt{\log t / n_t(k)}$ quantifies the uncertainty around the empirical average, and decreases as we collect more samples, i.e., as $n_t(k)$ increases. The larger this second term for an action k, i.e., the uncertainty of its performance, the higher the probability of choosing that arm, meaning that we need more samples to have a good estimate of its related reward. This principle is also known as *"optimism in the face of uncertainty."*

Thompson Sampling Agent (TS-A) The agent adopts a *Bayesian inference* approach to identify the most promising arms. In particular, TS-A builds a distribution for each reward, thus modeling not only its mean, but the whole statistics [214]. Given that our problem includes a binary reward $\{0, 1\}$ behind each arm, it is quite natural to model the rewards according to a Bernoulli distribution, which is parameterized by the success probability vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$, where μ_k represents the average unknown reward behind arm $k \in \mathcal{K}$. Following the Bayesian framework, parameter μ_k of arm k is modeled as a Beta(α_k, β_k) random variable, where α_k counts the number of successful transmissions after playing arm k, and β_k represents the number of collisions. Therefore, the mean of μ_k is equal to $\alpha_k/(\alpha_k + \beta_k)$. The Beta distribution parameters are initialized to $\{\alpha_k = 1, \beta_k = 1\}$ for all $k \in \{1, \ldots, K\}$.

As the TS-A collects more data, α_k and β_k are updated accordingly, inducing biased probabilities for the different arms. These informed distributions are also called *posterior probabilities*, in Bayesian parlance. Whenever the agent makes a decision, i.e., it chooses a physical channel based on the probability of that channel not being accessed by other agents in that time interval, it samples a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$, and plays the arm k^* such that $k^* = \text{argmax}_k\{\mu_k\}$. This algorithm is known as the Thompson Sampling (TS) algorithm [214]

Neural Agent (NA) The NA is equipped with a small-size neural network (NN) used to represent its decision policy. In particular, the agent receives, as an input, context information $s_t \in \mathcal{S}$ from the environment, thus the problem is formulated as a contextual MAB, i.e., the average reward depends on the played arm $k_{n,t}$, and on the state $s_{n,t}$ [148]. The NN input represents the feedback on the results of the last transmission attempt, broadcast by the gNB. As such, the input data is a vector of $K + 1$ entries: the first K values are the results of the transmission attempts in the K orthogonal channels, whereas the last value indicates whether it is a first-time transmission or a re-transmission. Again, the 0/1 reward given to failed/successful transmission, respectively, is used by the NA to optimize the NN parameters, and maximize the given rewards. The model is an adaptation of that in [51].

Remark. The UCB and TS algorithms exhibit good theoretical properties in terms of convergence time to optimal strategies, as long as some critical assumptions are satisfied [148]:

1. The rewards behind each action need to exhibit a sub-Gaussian distribution. Any distribution with limited support has this property, which is also verified in our setting.

2. The reward samples after playing action k are i.i.d. This assumption is more critical in real scenarios, and in particular in our problem. In fact, each agent interacts with many other devices, and so the rewards depend on the actions of the other agents, which are continuously learning and changing their decision schemes. This leads to highly non-stationary environments, meaning that the reward distribution may change over time. However, empirical results show that state-of-the-art MAB algorithms can still be applied even though the stationarity assumption for the rewards is not satisfied [49].

In Section 4.1.4 we compare the performance of the MAB agents presented above, and provide guidelines towards the best schemes to satisfy URLLC requirements for IIoT.

4.1.4 Performance Evaluation

In this section, after introducing our simulation setup, we evaluate the performance of the proposed distributed resource allocation scheme implementing one of the MAB agents described in Section 4.1.3, in different IIoT scenarios.

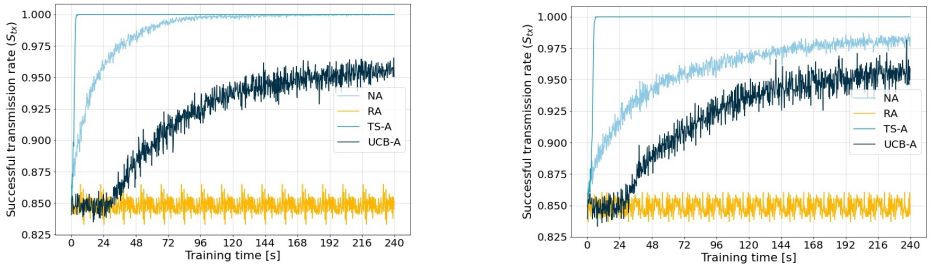
Simulation Setup

End machines transmit at frequency $f_c = 3.5$ GHz and with a bandwidth of $B = 20$ MHz. The subcarrier spacing is set to $\Delta f = 30$ KHz (i.e., 3GPP NR numerology 1), which results in $K = 55$ orthogonal channels, and an OFDM symbols duration of $T_{\text{OFDM}} \simeq 35.675\,\mu$s [204]. With an SU of 7 OFDM symbols, we get an SU duration of $T_{SU} \simeq 0.25$ ms. We assume that, whenever a packet is to be sent, it can be transmitted within one SU. If two or more UEs select the same UL channel for transmission in the same SU, we consider those packets to be lost (due to a collision event). Assuming that the gNB feedback (informing about the collision) is received within the current SU, the retransmission can be scheduled in the subsequent SU.

The factory floor is characterized according to the 5G-ACIA "Motion Control" scenario, as described in [12]. Hence, the geometry is modeled as a parallelepiped of length $\ell = 15$ m, width $w = 15$ m, and height $h = 3$ m, and machines are randomly and uniformly distributed inside the factory. The gNB is located at the center of the ceiling, and communicates with power $P_{\text{TX,DL}} = 30$ dBm. The transmit power of the UEs is set to $P_{\text{TX,UL}} \in \{8, 10, 23\}$ dBm. Also, we consider omnidirectional transmissions, therefore the antenna gain is fixed to $G = 1$ for both the UEs and the gNB. The channel model is based on the 3GPP Indoor Factory (InF) scenario [8], where UEs are assumed to communicate in Non-Line-of-Sight (NLOS) if the joining line between the UE's and the gNB's centers intersects one or more machines.

In our simulations, the traffic can be either *periodic* or *quasi-periodic*. In the first case, packets are generated at constant periodicity τ. In the second case, the application still generates packets with periodicity τ, upon which a random component t_{off} of $\{-2, -1, 0, +1, +2\}$ OFDM symbols is added.

The performance of the different MAB agents' policies is assessed in terms of *successful transmission rate* S_{TX}, which indicates the ratio between the successfully received packets and the



(a) Periodic traffic.

(b) Quasi-periodic traffic

Figure 4.2: S_{TX} vs. the training time, for different MAB agents, with periodic and quasi-periodic traffic, $\tau = 1.5$, and $N = 50$.

total number of attempts within one SU, averaged over 1 000 steps, as a function of the traffic periodicity τ, the number of UEs N, and the UL transmission power $P_{\text{TX,UL}}$. Notice that S_{TX} is inversely proportional to the number of re-transmissions and, as such, represents well the theoretical rewards $r_{n,t}$ of the MAB agents.

Numerical Results

Impact of the training. In Fig. 4.2 we analyzed the training curve of the agents with periodic and quasi-periodic traffic, with a periodicity $\tau = 1.5$ ms, and considering $N = 50$ UEs in the system, for a total training time of $T = 240$ s. For the periodic case, we observe from Fig. 4.2a that TS-A is the best performing agent. In particular, the TS agents are able to learn their optimal strategy, achieving zero collisions (i.e., $S_{TX} = 1$, our target for URLLC) in a very short training time (< 10 s). NA achieves a similar performance to that of TS-A, though after a longer training process. This is due to the fact that NA needs more interactions with the system to optimize the network parameters, thus slowing down the training phase. For UCB-A, the exploration parameter c in Eq. (4.1) was set to 2, as it showed the most stable configurations in our experiments. Still, it results in an even slower convergence compared to NA, due to the fact that it struggles to achieve coordination. Also, UCB-A presents significant oscillations over time, due to the impact of collisions and retransmissions. As expected, RA (our baseline) performs poorly, and there is no improvement over time, as feedback signals are not exploited by the algorithm to adjust the access scheme.

For the quasi-periodic case, we observe from Fig. 4.2b that TS-A presents again the best performance despite the more complex scenario, converging to zero collisions within 15 s. Now, NA no longer achieves perfect convergence within the training time, suggesting that it cannot work well in non-stationary multi-agent scenarios, or deal with non-deterministic traffic requests. However, we believe that, with a better tuned training process, and with more relevant context information as input, the final performance would reasonably improve. Finally, UCB-A and RA perform similarly to the case of periodic traffic.

Impact of the number of users. In Fig. 4.3 we evaluate the performance of the MAB agents as a function of $N \in \{25, 50, 75, 100\}$. In particular, we studied the statistics of the successful

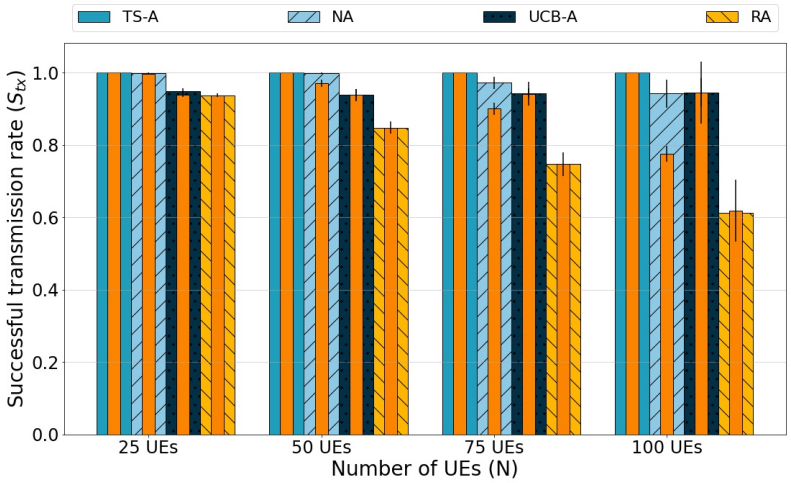


Figure 4.3: $S_{TX} \pm$ one standard deviation vs. N for different MAB agents, after a training time of 60 s, with $\tau = 1.5$ ms, with periodic (wide bars) and quasi-periodic (narrow bars) traffic.

transmission rate S_{TX} after 60 s of training, where again the total training time is set to $T = 240$ s. First, we observe that TS-A converges to the optimal scheme (i.e., $S_{TX} = 1$) within 60 s in all configurations, thus achieving coordination without communication even in dense ($N = 100$) networks. Second, NA outperforms UCB-A with periodic traffic, but suffers with quasi-periodic traffic: notably, S_{TX} decreases by 10% in the quasi-periodic case, for $N = 100$. This is due to the fact that NA implements and exploits an NN to optimize its decisions, thus the learning phase can take more time in the most complex scenarios. Interestingly, compared to other agents, UCB-A's performance is less sensitive to N, and eventually outperforms NA's approach in the most crowded scenarios. On the downside, it exhibits wider oscillations, i.e., higher standard deviation in Fig. 4.3, an indication of a less stable behavior of the agent in non-stationary environments.

Impact of the traffic periodicity. Fig. 4.4 explores the effect of the traffic periodicity τ on the successful transmission rate S_{TX}. As expected, the more aggressive the traffic, the more difficult for the agents to achieve convergence, which is also highlighted by the increased standard deviation in all MAB configurations. Again, TS-A is the best agent, and can converge to the optimal scheme regardless of the value of τ. Eventually, NA is also able to achieve zero collisions (i.e., $S_{TX} = 1$) when $\tau = 5$ ms in case of periodic traffic. Even the RA approach (our baseline) achieves a successful transmission rate of around 0.9 as τ grows, i.e., considering less bandwidth-hungry applications, thanks to the lower collision probability as the contention on the channel becomes less intense. Notably, UCB-A is the only method that improves the average accuracy as τ decreases: the shorter traffic periodicity implies more transmission attempts within the training time, which in turn provides more data to the agent to optimize its decisions. However, oscillations become significant when $\tau = 1.5$ ms.

Impact of the UL transmission power. IIoT devices, such as industrial sensors, may be subject to battery lifetime constraints. In light of this, we studied the impact of the UL

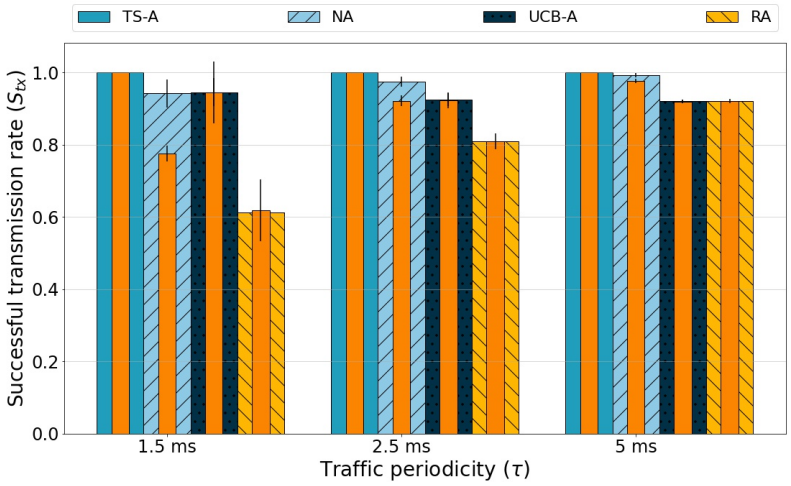


Figure 4.4: S_{TX} $\pm$ one standard deviation vs. τ for different MAB agents, after a training time of 60 s, with $N = 100$, with periodic (wide bars) and quasi-periodic (narrow bars) traffic.

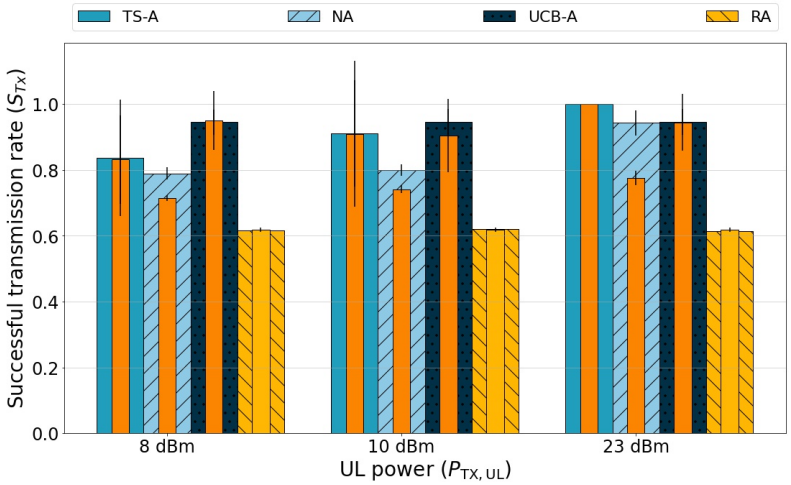
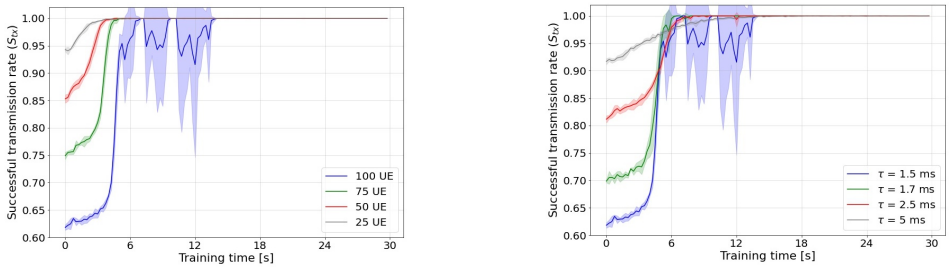


Figure 4.5: S_{TX} $\pm$ one standard deviation vs. $P_{\mathrm{TX,UL}}$ for different MAB agents, after a training time of 60 s, with $N = 100$ and $\tau = 1.5$ ms, with periodic (wide bars) and quasi-periodic (narrow bars) traffic.

transmission power $P_{\mathrm{TX,UL}} \in \{8, 10, 23\}$ dBm on the MAB convergence. While decreasing $P_{\mathrm{TX,UL}}$ promotes energy savings and mitigates interference, it may also lead to communication outage when the Signal to Interference plus Noise Ratio (SINR) goes below a pre-defined sensitivity threshold, set to -5 dB in our simulations. In Fig. 4.5, with $P_{\mathrm{TX,UL}} = 23$ dBm, the outage probability is very small, leading to $S_{TX} \approx 1$ in most configurations (if convergence is achieved). As $P_{\mathrm{TX,UL}}$ starts decreasing, outage events, besides collisions, lead to additional packet losses, and to a more complex environment. Unlike TS-A and NA, UCB-A is less sensitive to this effect. The



(a) (b)

Figure 4.6: S_{TX} vs. the training time and as a function of N (a) and τ (b), for TS-A with periodic traffic, and $\tau = 1.5$ ms. The curves report mean $\pm$ standard deviation over the simulation runs.

reasons are twofold. On one side, NA converges slowly, and is more exposed to retransmissions. At the same time, TS-A converges quickly to a specific solution, meaning that unpredictable outage events may break the environment statistics underlying the TS algorithm, and lead to unexpected negative feedback from the gNB. On the contrary, UCB-A initially explores more, and can better adapt to new configurations in more dynamic scenarios. When $P_{\text{TX,UL}} = 8$ dBm, UCB-A is the best performing agent, and achieves +16% S_{TX} compared to TS-A.

TS-A performance. In view of the above results, we further analyzed TS-A's convergence time to the optimal solution (where no collisions are experienced) as a function of (i) the number of users N, and (ii) the traffic periodicity τ. In Fig. 4.6a, we observe that, as N increases, the TS algorithm takes more time to converge to the best solution, as expected. Notably, the curve with $N = 100$ presents the highest variance, due to the fact that many users are learning an individual policy, leading to a highly non-stationary environment.

In Fig. 4.6b, we see that when τ decreases the convergence time grows accordingly, even though the gap among different configurations is relatively small (convergence is achieved after ~ 8 s). This is due to the fact that, on the one hand, when the traffic periodicity is short, the problem becomes more complex, as more packets have to be allocated. On the other hand, the agents receive more feedback signals within the same time interval, thus leveraging more data for the training.

Final remarks

Our initial experiments confirm that there exists a MAB configuration for which distributed resource allocation can achieve zero collisions in low latency, i.e., without gNB scheduling grants, thus supporting URLLC.

In particular, TS-A is the best performing approach in case of both dense systems and aggressive aperiodic traffic (where conventional semi-persistent/grant-free NR schedulers may fail). Consequently, our experiments suggest that the Bayesian formulation, together with the exploration strategy of TS, are good starting points to build distributed resource allocation in real IIoT environments, reducing the latency introduced by centralized protocols. Interestingly, UCB-A

works well in complex scenarios, or when UEs communicate with limited power, thus supporting energy efficiency at the expense of some collisions. Moreover, the superior performance in terms of S_{TX} of the MAB schemes against RA shows that machine learning can dramatically reduce, if not completely eliminate, the burden of re-transmissions introduced by 5G-NR-like grant-free access scheduling schemes [185]. However, distributed resource allocation requires longer training time before convergence, which in real IIoT systems may not be negligible. Still, the training could be run offline, which does not affect the real-time performance of the system (it can be executed when the machine is turned off, e.g., during the calibration of the electro-mechanical processes, or before the service is activated); once active, the service can run rapidly and without significant computational overhead.

Moreover, our analysis evaluates the training time when the system starts the optimization process from scratch: faster adaptation can be achieved if the system faces limited changes with respect to the initial training scenario, e.g., some components join or leave the system. Nevertheless, the trained model still requires retraining when data distributions have deviated significantly from those of the original training set, which involves additional overhead [144]. This motivates further explorations in the case of more dynamic systems, that will be carried out as part of our future work.

4.1.5 Conclusions and Future Work

We studied the design of user-centric (rather than gNB-centric) distributed (rather than centralized) resource allocation in IIoT scenarios. This approach does not involve scheduling grants to be disseminated before UL transmissions, and is thus positioned to better support URLLC compared to conventional scheduling methods. We explored different state-of-the-art MAB agents, for the first time applied to the context of URLLC for IIoT, and identified TS-A as the best performing implementation, achieving zero collisions in our experiments. TS-A scales well with the number of users in the system compared to other MAB methods, and still achieves perfect accuracy even considering aperiodic traffic. Notably, UCB-A showed superior performance when the UEs communicate with low power, despite some collision events.

4.2 DISNETS: a DIStributed NEural linear Thompson Sampling framework to achieve URLLC in IIoT

In the previous section we've seen a preliminary analysis on the application of state-of-the-art MAB algorithms to the distributed resources allocation problems in wireless networks, with the aim of reducing the latency needed to transmit data in an IIoT environment. This work extends the analyses by considering a more complex but accurate system model, and by proposing a novel proposal which augments the linear TS with the adoption of a deep neural network (DNN)[190].

4.2.1 Introduction

As we have seen in Section 4.1, IIoT poses strict communication requirements to achieve almost real-time coordination, control, and sensing [151]. Specifically, these requirements translate into latency (less than 1 ms in the radio part) and reliability (up to 99.99999%) constraints, thus calling for Ultra-Reliable and Low-Latency Communications (URLLC) [10, 7]. However, the 5th generation (5G) standard is unlikely to provide resource allocation in short time, mainly due to the intrinsic limitations of current channel access schemes [156, 74]. Notably, the 3rd Generation Partnership Project (3GPP) New Radio (NR) specifications for 5G networks [4] support three options to allocate uplink (UL) resources: grant-based scheduling (GBS), semi-persistent scheduling (SPS), and grant-free scheduling (GFS) [158]. GBS [1] is fully centralized, and requires: (i) the User Equipments (UEs) to use the Physical Uplink Control Channel (PUCCH) to ask the uplink scheduler for being scheduled; (ii) the gNB to communicate via the Physical Downlink Control Channel (PDCCH) to the UEs which resources can be used for transmission; (iii) the UEs to transmit their data blocks through the Physical Uplink Shared Channel (PUSCH); and (iv) the gNB to provide the communication acknowledgment via Hybrid Automatic Repeat reQuest (HARQ). This procedure requires at least two Round Trip Times (RTTs) from when data arrives in the buffer until it can be properly scheduled, which may prohibitively increase the communication delay. SPS [6] is also fully centralized, but permits the Next Generation Node B (gNB) to pre-allocate radio resources without explicit scheduling requests and grants from/to the UEs, thus reducing the latency. However, SPS works under the assumption of periodic/predictable traffic, which is not always verified in the Industrial Internet of Things (IIoT) scenario, and may cause large and systematic delays in case of errors in those predictions. On the other extreme, GFS [165] is fully distributed, and the UEs autonomously choose radio resources to be used for transmission, thereby eliminating the need to wait for scheduling grants. On the downside, uncoordinated resource allocation may lead to potentially many collisions, and trigger re-transmissions accordingly, which again poses additional latency concerns.

In this context, machine learning (ML) has emerged as a promising tool to optimize network performance, including minimizing latency during resource allocation. Still, most of the literature focuses on centralized and downlink algorithms, e.g., in [135], which however are not scalable as the density of the network increases. In the area of distributed learning, multi-armed bandit (MAB) algorithms [226], and especially Linear Thompson Sampling (LTS) [212], gained popularity to address the problem of resource allocation. However, these schemes are often too easy to model complex network dynamics, and work with the assumption of linear dependency of data [194]. A promising attempt to overcome this limitation was made with the Neural Linear Thompson Sampling (NLTS) algorithm [210], which still assumes that the ML agent can only play single actions, i.e., UEs transmit through one single orthogonal channel, which may increase the latency beyond URLLC requirements.

To solve these issues, in this work we propose a new distributed framework for resource allocation called DIStributed combinatorial NEural linear Thompson Sampling (DISNETS), which is built upon two cardinal principles. First, it consists of a UE-centric architecture in which resource

allocation decisions are made by the local UEs, "disaggregated" from the network, and without pre-defined scheduling requests and/or grants. Second, UEs rely on ML to optimize resource allocation accordingly, which allows to minimize the probability of collisions and reduce the latency due to re-transmissions. To this aim, our contributions are the following:

- We formalize the problem of distributed resource allocation as a Multi-Agent Contextual Combinatorial Multi-Armed Bandit (MA-CC-MAB) problem, where UEs autonomously choose the physical resources to use for transmission. The problem is solved using DISNETS, built on top of the NLTS algorithm [210], which combines deep neural network (DNN) and LTS to optimize network operations. Specifically, the original NLTS implementation is extended into the proposed DISNETS solution by allowing agents to take more than one action at each scheduling opportunity, i.e., using multiple orthogonal channels in parallel in the same scheduling opportunity, which is important to provide URLLC.
- We propose the design and structure of a new control signaling scheme, referred to as Feedback Control Information (FCI), and used by the UEs to train and learn how to allocate resources using DISNETS. The structure of the FCI is similar to that of the Downlink Control Information (DCI) signal, which is currently used in 5G NR to enable centralized scheduling [185].
- We apply DISNETS to the context of URLLC in IIoT environments. As such, we propose a new ad-hoc traffic model in which industrial machines and users in a production line activate and generate traffic, respectively, based on some temporal and spatial correlations. This approach promotes more realistic, IIoT-specific simulations.
- We validate DISNETS through end-to-end (E2E) simulations in terms of latency and reliability, against 5G NR GBS ans SPS baselines for resource allocation, and GFS based on random access. Simulation results are as a function of the number of UEs in the network, the traffic configuration, and some other IIoT-specific system parameters. We show that DISNETS achieves faster and more accurate resource allocation than its competitors, also considering aperiodic and unpredictable traffic.

4.2.2 System Model

In this section we present our system model. Specifically, we describe our factory layout and scenario, the channel model, the traffic model to characterize IIoT-specific interactions between machines, end users, and the underlying factory geometry and functionalitie. In the end, we provide the E2E latency and reliability models.

Scenario

Factory floor We consider a limited geographical area within an indoor factory floor, modeled as a parallelepiped of length l, width w, and height h, as reported in [14]. Then, M industrial machines are grouped into W different production lines (each of which models an underlying

industrial process), and are connected to a Standalone Non-Public Network (SNPN), i.e., a 5G remote and private network with a reserved Radio Access Network (RAN) and 5G Core (5GC) [11]. For example, Fig. 4.7 illustrates an example with $W = 2$ production lines and $M = 7$ machines. Machines are modeled as cubes of size S, and deployed across the factory floor according to a uniform distribution, ensuring a given inter-machine distance D (relative to the centers of machines) and a minimum number of machines $M_{\min}$.

Onboard the machines, N UEs are distributed following the same method, at a maximum height S, and generate traffic according to pre-defined patterns (more on this later). Moreover, obstacles act as obstructions between the UEs and the gNB.

Resource allocation We consider uplink communication, i.e., from the N UEs to the Controller/Master (C/M), that is a remote entity monitoring and controlling the machines from the 5GC through a gNB. The total available bandwidth B is split into K orthogonal channels where, according to the 3GPP nomenclature, an orthogonal channel consists of 12 Orthogonal Frequency Division Multiplexing (OFDM) subcarriers. Time is also discretized into Scheduling Units (SUs) whose duration is of 7 OFDM symbols. Then, a Resource Block (RB) is defined as the minimum physical resource unit that can be allocated for data transmission, and consists of one orthogonal channel in frequency, and one SU in time. Within one SU, the first 4 OFDM symbols are dedicated to the PUSCH, used by the UEs to transmit data, and the last 2 OFDM symbols are used by the gNB to convey the FCI, as described in Section 4.2.3.

Data transmission Whenever a UE has new data packets to send, it can use multiple RBs choosing different orthogonal channels within the same SU. The assumption in our system model is that, whenever two or more UEs use the same orthogonal channels in the same SU, i.e., the same set of RBs, they create collision, and we assume data packets to be lost, i.e., they cannot be detected by the gNB.

Channel Model

The channel is characterized based on the Indoor Factory (InF) model for IIoT networks [8, Table 7.2-4]. Specifically, the 3GPP identifies several InF scenarios depending on the density of obstacles and the location of the UEs with respect to the gNB. Among these, in this work we selected the most representative 3GPP InF scenario based on the 5G-ACIA factory layout and geometry described in [11].

The path loss depends on the Line-of-Sight (LOS) or Non-Line-of-Sight (NLOS) condition of the channel. In this regard, the 3GPP provides an expression for the LOS probability in [8]. However, we do not adopt a statistical model to discriminate between LOS and NLOS propagation. On the contrary, we implement a geometry-based approach that checks whether the joining line between the UE's and the gNB's centers intersects one or more obstacles. If so, the UE is considered in NLOS, otherwise it is in LOS. Then, the quality of the received signal is assessed in terms of the

Signal to Noise Ratio (SNR), which is defined as

$$\text{SNR} = \frac{P_{\text{TX}} \cdot G_{\text{UE}} \cdot G_{\text{gNB}}}{PL \cdot P_N}, \tag{4.2}$$

where P_{TX} is the transmit power, G_{UE} and G_{gNB} are the antenna gains at the UE and the gNB, respectively, PL is the path loss, and P_N is the Additive White Gaussian Noise (AWGN) noise power. The latter is computed as $k_B \cdot T \cdot B$, where k_B is the Boltzmann constant, T_B is the system noise temperature (in K), and B is the total bandwidth (in Hz) at the gNB. The SNR is used to check whether a data block is correctly decoded, i.e., if the SNR is above a given threshold SNR_{th}, and to determine the modulation order to be used for data transmission according to [3].

The Spatio-Temporal Correlated Traffic Model

In most literature work, uplink traffic is assumed either periodic (i.e., UEs generate data at predefined time intervals) or totally aperiodic (i.e., UEs generate data with a variable periodicity), and machines in the factory floor can generate packets simultaneously. In this work, we propose an alternative traffic model where data packets are generated according to pre-defined statistics, to better characterize IIoT interactions. In this model, machines within a production line are sequentially activated, emulating the workflow of the corresponding industrial process. Then, UEs onboard active machines produce data traffic (either periodic or aperiodic [13]) for an entire "activation period" of duration τ_a, after which another machine in the current production line will activate.

As such, the traffic model accounts for both spatial and temporal correlation, as illustrated in Fig. 4.7. Notably, it is possible to identify two different types of correlation.

Inter-machine correlation It refers to the way the machines activate. Specifically, when an event occurs, one machine per production line is activated. Then, after the activation period, the next machine activates following the flow of the production line, at a sequence that depends on the factory geometry and the number of machines and lines.

Intra-machine correlation It refers to the way UEs onboard the active machines generate packets. After one machine per line is activated, the UEs associated with those machines activate too. Then, active UEs generate a flow of packets according to some statistics, e.g., in terms of the inter-packet interval and/or the packet size, until some new machines in the production line activate. We consider:

- *Periodic traffic*, if packets are generated at constant periodicity τ [13]. For example, UEs periodically measure and report physical parameters (e.g., temperature, pressure, radiation) from the production process.
- *Uniformly aperiodic traffic*, if the inter-packet interval τ is modeled as a uniform random variable in $[t_{min}, t_{max}]$, $\forall n \in \mathcal{N}$, where $\mathcal{N}$ is the set of UEs [13]. For example, UEs make unscheduled aperiodic transmissions in case abnormal measurements are detected.

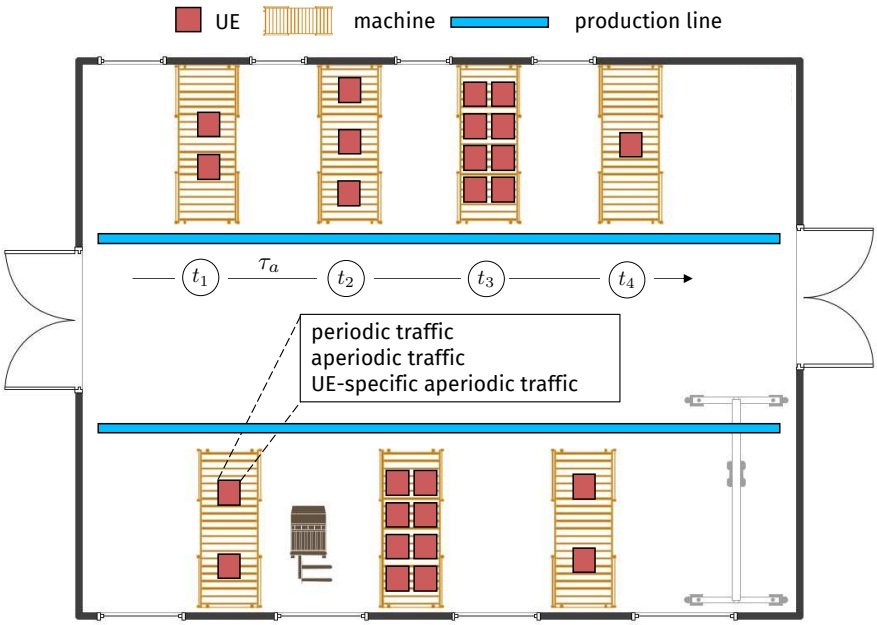


Figure 4.7: Factory floor layout (with $W = 2$, $M = 7$, and $N = 18$) and traffic correlation. Specifically, machines in each production line are correlated, and activate according to a specific sequence on the production line, i.e., toward the right or the left. At t_1, $W = 2$ machines (i.e., one per production line) activate, and the corresponding UEs onboard the active machines start sending data as periodic, aperiodic, or UE-specific aperiodic traffic. At $t_2 = t_1 + \tau_a$, these machines shut down and the next activation begins.

- *UE-specific aperiodic traffic*, in which the extreme values t_{min} and t_{max} are UE-dependent parameters. Specifically, we now assume that UE_n, $\forall n \in \mathcal{N}$, generates with probability one another packet in the interval $[t^n_{min}, t^n_{max}]$, with t^n_{min} and t^n_{max} modeled as uniform random variables within the intervals $[t_{min}, t_{max}]$ and $[t^n_{min}, t_{max}]$, respectively. Notably, we now have $t_{min} \leq t^n_{min} \leq t^n_{max} \leq t_{max}$. The rationale behind this new traffic model is that, in real-world factories, some sensors/UEs may control different parts or mechanisms of the same inter-machine process, thus activating with statistics that depend on their roles or position, and so are UE-specific.

Latency and Reliability Models

Latency We require our system to minimize the E2E latency in uplink, which is defined as the time from when one packet is generated at the UE's application to when the same packet is successfully received by the C/M. Specifically, the E2E latency L of a packet is computed as:

$$L = T_P + T_{\text{RAN}} + T_{\text{TX}} + \tau_P + T_{\text{DAS}} + \tau_F + T_{\text{gNB}} + T_{\text{CN}}, \tag{4.3}$$

where, based on the analysis in [74]:

- T_P is the time for the UE to create the data packet, i.e., to add headers across the 5G protocol stack;
- T_{RAN} is the time between the generation of the data packet at the Physical (PHY) layer and the packet transmission, which depends on the scheduling algorithm;
- T_{TX} is the transmission time;
- τ_P is the propagation time from the UE to the fronthaul of the gNB, i.e., the Distributed Antenna System (DAS);
- T_{DAS} is the time for the DAS to process the received data packet, and to send it to the gNB;
- τ_F is the time for the signal to travel from the DAS to the gNB, generally through a high-capacity optical fiber;
- T_{gNB} is the time for the gNB to process the received data block, and to send it to the C/M;
- T_{CN} is the delay introduced by the 5GC, that is the time for the message to reach the C/M from the gNB.

Finally, we denote as $\bar{L}$ the average E2E latency, averaged over the data packets generated by the UEs within the simulation time T_S, and over the number of UEs.

Reliability We also require our system to operate with high reliability. In this sense, we define a reliability metric $\eta_t(L_{\text{th}})$ as the empirical probability that the E2E latency of a packet is below a pre-defined threshold L_{th} during SU_t, and $\bar{\eta}_t(L_{\text{th}})$ is the empirical average of $\eta_t(L_{\text{th}})$ within the simulation time T_S.

4.2.3 Problem Formulation

The aim of our work is to minimize T_{RAN} in Eq. (4.3), which dominates the E2E latency, and depends on the underlying resource allocation procedure. As mentioned in Section 4.2.1, standard 5G NR protocols mainly adopt either centralized scheduling at the gNB (i.e., GBS and SPS), which introduces delays due to the rigidity of the resource allocation scheme with respect to the traffic generation process, or distributed scheduling (i.e., GFS), which may result in collisions. In turn, we propose DISNETS, a new scheduling framework that combines the benefits of the two: on one side, resource allocation is decentralized, in the sense that UEs autonomously decide how to allocate resources without significant interactions with the gNB, which eliminates the waiting time to receive scheduling grants; at the same time, UEs exploit ML to optimze scheduling decisions based on traffic correlations, which may reduce the probability of collision. Our research problem is formulated as a MA-CC-MAB problem, as described below.

The CC-MAB Problem

The problem formulation is built on top of the Contextual Combinatorial Multi-Armed Bandit (CC-MAB) framework [61]. Specifically, every time $\text{UE}_n \in \mathcal{N}$, i.e., an agent, has data to send in SU_t, it will autonomously choose the physical resources to be used for transmission. The total available bandwidth is split into K orthogonal channels, and we denote with $\mathcal{K} = \{1, 2, \ldots, K\}$ the set of channels. In CC-MAB parlance, the K orthogonal channels are the K feasible actions that can be chosen by the agent. To take an action, each agent can rely on side information, i.e., the context $s_t \in \mathcal{S}$, that describes the state of the environment, i.e., the wireless network, in SU_t, which we model as a random variable sampled according to the system's probability P_S. Given the context s_t and the action k_t chosen by the agent in SU_t, the environment returns a reward $r_t \in [-1, 1]$ according to the probability $P_R(s_t, k_t)$, which reflects the probability that data transmission using channel k_t in context s_t was successful. Specifically, $r_t = -1$ if collisions happen, otherwise it is proportional to the number of transmitted bits if the transmission is successful (see Eq. (4.10) for further details). Then, $\mu(s, k)$ is the average reward with respect to the distribution $P_R(s, k)$ $\forall s \in \mathcal{S}, \forall k \in \mathcal{K}$.

In our framework, the CC-MAB problem is extended by allowing agents to take more than one action in each SU, i.e., using multiple orthogonal channels in parallel in the same SU, which is important to provide URLLC. Therefore, we define a super-action $\theta_t \subset \mathcal{K}$ as a set of actions in SU_t, so θ_t is an element of the super-set Θ of $\mathcal{K}$, i.e., the set of all possible subsets of $\mathcal{K}$. The reward r_t is then sampled according to $P_R(s_t, \theta_t)$. Interestingly, we can exploit the structure of the environment to assume that

$$\mu(s, \theta) \sim \sum_{k \in \theta} \mu(s, k), \quad \forall s \in \mathcal{S},\ \forall k \in \mathcal{K}, \tag{4.4}$$

i.e., the average reward relative to super-action θ is proportional to the sum of the average rewards of the single actions.

To choose the super-action in SU_t, the agent employs a policy $\pi_t : \mathcal{H}^{t-1} \times \mathcal{S} \to \Phi(\Theta)$, which is a map from the history $H_t = \{(s_1, \theta_1, r_1), \ldots, (s_{t-1}, \theta_{t-1}, r_{t-1})\} \in \mathcal{H}^{t-1}$ of previous contexts, actions, and rewards, to a probability distribution over the set of feasible super-actions Θ. Given an horizon T, the goal of the agent is to find the policy π^* that maximizes the expected sum of rewards over time, i.e.,

$$\pi^* = \underset{\pi}{\text{argmax}}\, \mathbb{E}\left[\sum_{t=1}^{T} \mu(s_t, \theta_t)\right], \tag{4.5}$$

where the expectation is taken with respect to the distributions of the environments P_R and P_S, and the agent's policy π_t, used to sample super-actions according to $\theta_t \sim \pi_t(s_t)$.

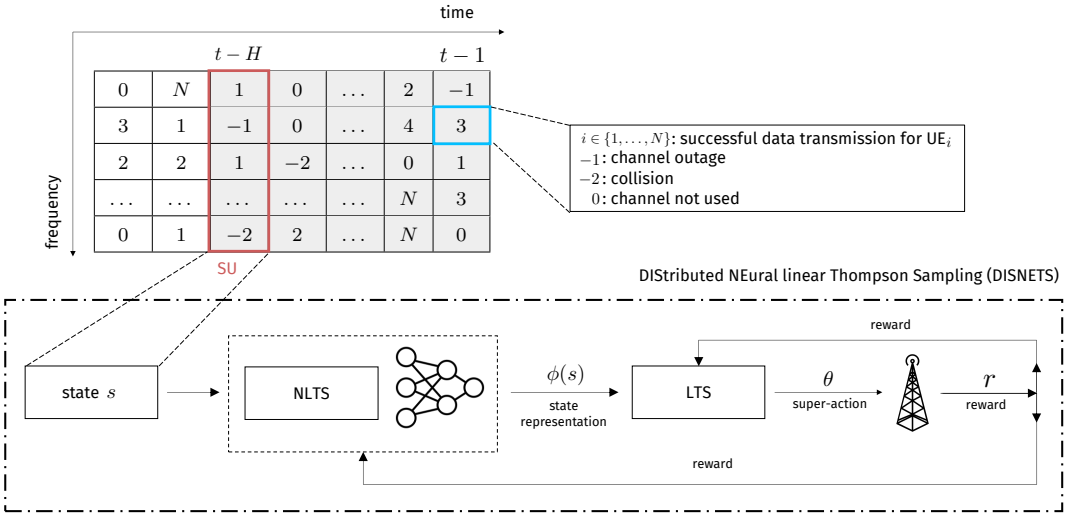


Figure 4.8: Schematic representation of DISNETS. The framework consists of (i) the state/context s, (ii) an NLTS module to provide the non-linear representation of the context $\phi_\omega(s)$, (iii) an LTS module to choose a super-action $\theta \in \mathcal{K}$ corresponding to the set of orthogonal channels to use to transmit data, (iv) the reward r (incorporated within the FCI) to update the NLTS and LTS parameters.

The MA-CC-MAB Problem

In our work we further extend the CC-MAB problem, and formulate a new MA-CC-MAB problem where the reward depends on the other agents' actions in SU_t. Therefore, the reward $r_{n,t}$, the context $s_{n,t} \in \mathcal{S}$, and the policy $\pi_{n,t}$ are now a function of n. This is critical to support more accurate resource allocation in a distributed scenario, where there is no or little interaction among the UEs.

A collision event is expressed by the random variable $\chi_t(s,k) \in \{0,1\}$, which is equal to 1 when collision(s) happen in the orthogonal channel k in SU_t, and 0 otherwise. As such, it is modeled as a Bernoulli random variable of parameter $\varphi_t(s_n, \boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t = (\theta_{1,t}, \dots, \theta_{N,t})$ is the vector of super-actions of all the N agents.[1] By model assumption, whenever two agents n and n' play super-actions $\theta_{n,t}$ and $\theta_{n',t}$, respectively, we have that $\chi_t(s,k) = 1$, $\forall s \in \mathcal{S}$, $\forall k \in \theta_{n,t} \cap \theta_{n',t}$, as collisions occur. Notice that agents make decisions based only on local data, so the structure of vector $\boldsymbol{\theta}_t$ is unknown a priori. As such, agents can just rely on the local context $s_{n,t}$, and the statistical knowledge acquired through history $H_{n,t}$.

4.2.4 Proposed Solution: the DISNETS Framework

In this section we present the proposed solution to the MA-CC-MAB problem. The proposed framework implements (i) a new control signal called FCI, which conveys the information to

[1] Notice that $\chi_t(s,k)$ does not explicitly depend on n but has the same value for all the agents. However, the local context $s_{n,t}$ does, which makes $\chi_t(s,k)$ agent-dependent.

compute the reward $r_{n,t}$ and is used to generate the context $s_{n,t}$, and (ii) the DISNETS algorithm, which combines DNN and LTS to solve Eq. (4.5), and allows end users to autonomously allocate resources for uplink transmissions.

Context and FCI Implementation

As introduced in Section 4.2.3, the agents optimize policies $\pi_{n,t}$ based on the rewards $r_{n,t}$, $\forall n \in \mathcal{N}$. In our framework, the reward is based on the new FCI signal depicted in Fig. 4.8, which has a similar structure than the 5G NR DCI signal. The FCI, sent from the gNB to the UEs, is located in the last 2 OFDM symbols of each SU, and includes the transmission outcomes in that SU relative to each orthogonal channel. Based on the received power on each orthogonal channel, the gNB can distinguish among four outcomes, namely successful transmission for UE_i ($\text{FCI} = i$, $i \in \{1, \ldots, N\}$), outage ($\text{FCI} = -1$), collision ($\text{FCI} = -2$), or channel not used ($\text{FCI} = 0$).

Based on the FCI, the UEs can compute their local contexts $s_{n,t}$, $\forall n \in \mathcal{N}$, which are used to statistically describe the state of the system in SU_t. Specifically, the context $s_{n,t}$ aggregates the outcomes from H previous FCI signals, which provides a history of previous resource allocation decisions.

DISNETS Implementation

In this section we describe our DISNETS framework, represented in Fig. 4.8, to solve the MA-CC-MAB and support distributed resource allocation. DISNETS extends the NLTS algorithm, which is in turn based on the LTS algorithm.

Notation. For simplicity, we drop subscript n, with $n \in \{1, \ldots, N\}$, to represent the index of the agent/UE.

Linear Thompson Sampling (LTS) DISNETS is built on top of the LTS algorithm [19] to choose an action $k \in \mathcal{K}$, which returns the orthogonal channel that the UE can use to transmit data. LTS assumes that the average reward behind each action $k \in \mathcal{K}$ is a linear function of the context $s_t \in \mathcal{S}$, and of an unknown action parameter vector β_k, i.e., $\mu(s, k) = \langle s, \beta_k \rangle$. In this case, estimating the most accurate vector β_k, $\forall k \in \mathcal{K}$, turns out to be an online linear regression problem. The problem is online because the target values, i.e., the rewards associated to the pair (s, β_k), are observed by the agent as it interacts with the environment and takes actions, respectively, and are not available at the beginning of the training process, like in supervised learning.

To solve this online problem, LTS assumes that the rewards, given the context s and action k, are modeled as a Gaussian random variable $R(s, k)$,[2] i.e., $R(s, k) \sim \mathcal{N}\left(s^T \beta_k, \nu_k^2\right)$. The LTS algorithm maintains a distribution of the parameter vector of each action k at time t, i.e., $\beta_k(t)$.

[2]According to [19], the rewards are not required to be Gaussian to converge to the optimal actions, but their domains need to be bounded.

Therefore $P(\beta_k(t)) \propto \mathcal{N}\left(\hat{\beta}_k(t), \nu_k^2 \left(\Phi_k(0) + \Phi_k(t)\right)^{-1}\right)$, where

$$\Phi_k(0) = \lambda \cdot I_d; \tag{4.6}$$

$$\Phi_k(t) = \sum_{\tau=1}^{t-1} s_\tau s_\tau^T \cdot \mathbf{1}_{k_t=k}; \tag{4.7}$$

$$\hat{\beta}_k(t) = \left(\Phi_k(0) + \Phi_k(t)\right)^{-1} \sum_{\tau=1}^{t-1} s_\tau r_\tau \cdot \mathbf{1}_{k_t=a}. \tag{4.8}$$

and λ is a hyper-parameter governing the initial exploration. After observing context s_t and taking action k_t, the agent receives a reward r_t based on the FCI (described above), and updates the distribution of the parameter vector at time $t+1$ as

$$\begin{aligned} &P(\beta_k(t+1)|s_t, r_t) \\ &\propto P(r_t|s_t, \beta_k(t+1)) \cdot P(\beta_k(t+1)) \\ &\propto \mathcal{N}\left(\hat{\beta}_k(t+1), \nu_k^2(\Phi_k(0) + \Phi_k(t+1))^{-1}\right). \end{aligned} \tag{4.9}$$

Based on the posterior update rule in Eq. (4.9), the agent samples K vectors $\hat{\beta}_k$, $\forall k \in \mathcal{K}$, and plays action $k_t = \text{argmax}_{k\in\mathcal{K}} \langle s_t, \hat{\beta}_k \rangle$. The interesting property of LTS is that it automatically balances *exploration*, i.e., sampling random actions to explore their reward statistics, and *exploitation*, i.e., exploiting the knowledge collected in previous time slots to optimize future network decisions, and presents good empirical performance [194].

Neural Linear Thompson Sampling (NLTS) As discussed in the previous paragraph, LTS assumes a linear relation between the context and the average reward obtained by playing one specific action. However, linear relations are not complex enough to model real-world scenarios. This problem is usually solved using Deep Reinforcement Learning (DRL), which requires a long training phase. On the other hand, DNNs are potentially good candidates to model the many different relations between contexts and actions. On the downside, when using DNNs there are no closed form solutions for the posterior updates, like the one in Eq. (4.9). As such, different approximate solutions have been proposed in the literature.

In particular, the authors in [210] introduced a new algorithm, called NLTS, that models the non-linearity between contexts and rewards by assuming that the average reward $\mu(s,k)$ is equal to a linear combination of the action parameter vector β_k and a non-linear representation of the context $\phi(s)$, i.e., $\mu(s,k) = \langle \phi(s), \beta_k \rangle$. To do so, a DNN $f_\omega(s)$: $\mathcal{S} \to \mathcal{R}^K$, parameterized by the weights vector $\omega \in \Omega$, is trained to estimate the average reward for each action, where $\mathcal{R}$ is the range of the actions' reward, assumed to be the same for all the actions. The non-linear representation of the context $\phi(s)$, or $\phi_\omega(s)$ to illustrate the dependency from ω, is the output of the last hidden layer of the DNN, i.e., the input of the last layer, whose output is $f_\omega(s)$. Then, NLTS uses LTS to choose the orthogonal channel for the UEs to transmit data, working on the state representation $\phi_\omega(s)$ in place of s.

Algorithm 4.1 Neural Linear Thompson Sampling

Initialize $\Phi_k(0) = I_d$, $\hat{\beta}_k(0) = \beta_k(0) = 0$, $\psi_k = 0$

for $t \in 1, \ldots, T$
  Observe s_t and compute $z_t = \phi_\omega(s_t)$
  Sample $\nu_k(t)$, $\forall k \in \mathcal{K}$, from $\text{IG}(a_k(t), b_k(t))$
  Sample β_k, $\forall k \in \mathcal{K}$, from $\mathcal{N}(\hat{\beta}_k(t), \nu_k^2 \left(\Phi_k(0) + \Phi_k(t)\right)^{-1})$
  Play $k_t = \text{argmax}_{k \in \mathcal{K}} z_t^T \beta_k$
  Observe r_t and store (s_t, k_t, r_t) in the buffer
  Update action posterior (Eq. (4.9)), using context z_t
  Update noise posterior

  if t % O = 0
    Train f_ω with **SGD** using samples in the buffer
    Compute new z_t, and update LTS
  end if
end for

We highlight that the posteriors of LTS are updated based on Eq. (4.9) every time the agent makes a decision, while the parameters ω of the DNN are updated at fixed intervals of O steps.[3] Another feature introduced by NLTS is that the distribution of the noise variance of the reward ν_k is modeled using the Inverse Gamma distribution, i.e., $\nu_k(t) \sim \text{IG}(a_k(t), b_k(t))$. The pseudocode of NLTS is reported in Algorithm 4.1.

DISNETS (Proposed) As mentioned, we now enhance the NLTS algorithm to allow each agent to choose multiple actions, i.e., to use multiple orthogonal channels in the same SU as expected in real systems, which poses the MA-CC-MAB problem in Section 4.2.3. Intuitively, there are two main issues when applying basic NLTS to MA-CC-MAB: first, the super-action θ that maximizes the expected reward $\mu(s, \theta)$ is unknown a priori, as the effects of single actions can be combined in different ways to obtain the super-action's reward [258]; second, the complexity of combining super-actions increases exponentially with the number of single actions, which is not tractable.

We design the reward for a single action $k \in \mathcal{K}$ in SU_t as

$$r_t = \begin{cases} \bar{\rho}_k(t) & \text{if } \chi_t(s,k) = 0; \\ -1 & \text{if } \chi_t(s,k) = 1, \end{cases} \tag{4.10}$$

where $\bar{\rho}_k(t) \in [0, 1]$ represents the total number of bytes that can be sent during SU_t using channel k, normalized by the maximum number of bytes that can be sent when using the maximum modulation order, and $\chi_t(s, k) \in \{0, 1\}$ indicates whether a collision happens using channel k during SU_t. As such, the reward in Eq. (4.10) is a function of both the channel and the resource

[3] It is important to properly balance the optimization steps of the LTS module (performed at each interaction with the environment) and that of the NLTS module (performed every O interactions with the environment) of DISNETS to avoid training instabilities, as one module depends on the other).

allocation policies of all the agents.

Furthermore, based on the assumption in Eq. (4.4), we have that the average reward of super-action θ is the sum of the single average rewards, i.e.,

$$\mu(s,\theta) = \sum_{k\in\theta} \langle \phi_{\omega^*}(s), \beta_k \rangle, \ \forall s \in \mathcal{S}, \ \forall \theta \in \Theta, \tag{4.11}$$

where we assume there exists a DNN $\omega^* \in \Omega$ that provides the exact non-linear representation of the context s, $\forall s \in \mathcal{S}$. The reward in Eq. (4.11) gets to a particular case of MA-CC-MAB referred to as *matroid bandits* [145]. In this case, it is possible to estimate the average reward obtained by super-action θ as the sum of the estimated average rewards of all its base actions.

Based on the above introduction, we extend NLTS into the proposed DISNETS algorithm so that the agent can play super-action θ_t in SU_t based on the following criteria:

1. Sample K vectors $\{\hat{\beta}_k\}_{k=1}^{K}$ as described above;
2. Compute the non-linear representation of the context, i.e., $\phi_\omega(s_t)$, based on the context s_t at SU_t;
3. Take super-action $\theta_t = \{k \in \mathcal{K} \ : \ \phi_\omega(s_t)^T \beta_k > \epsilon\}$, i.e., the agent transmits using all the orthogonal channels whose estimated reward (which is an indication of the transmission data rate) is larger than ϵ.

Consequently, the average number of orthogonal channels that an agent can use is not constant, but rather learned and adjusted via DISNETS given the reward history and the context at time t. In addition, we further introduced a variance decaying factor γ to scale the sampling variance of the reward $\nu_k(t)$ by γ, in order to force DISNETS to become more deterministic as the training progresses.

4.2.5 Numerical Results

We now provide our simulation parameters, show the convergence performance of DISNETS, and we then evaluate the performance of DISNETS against some other benchmarks in terms of overhead, latency, and reliability.

Simulation Parameters

Simulation parameters are reported in Table 4.1.

System parameters The system operates with a carrier frequency of $f_c = 3.5$ GHz and a bandwidth of $B = 60$ MHz. We set 3GPP NR numerology 2 (i.e., a subcarrier spacing of $\Delta_f = 60$ KHz), which leads to 84 RBs [2]. For the latency in Eq. (4.3), according to the 5G standard specifications we assume that (i) the processing times T_P and T_{gNB} at the UEs and the gNB, respectively, are both equal to 7 OFDM symbols, (i.e., 116.9 μs for numerology 2), (ii) the propagation time τ_P is neglected because it can be compensated with an accurate timing advance technique (see [9]), and (iii) τ_F is also neglected due to its minor impact on L.

Table 4.1: Simulation parameters.

Parameter	Value
Carrier frequency (f_c)	3.5 GHz
Overall system bandwidth (B)	60 MHz
5G protocol stack header (H)	72 bytes [85]
Subcarrier spacing (Δf)	60 kHz
SNR threshold (SNR_{th})	-5 dB
Latency threshold (L_{th})	1 ms
Noise temperature (T_B)	290 K
Antenna gain ($G_{\text{UE}} = G_{\text{gNB}}$)	0 dB
UE (UL) transmit power ($P_{\text{TX,UL}}$)	23 dBm
gNB (DL) transmit power ($P_{\text{TX,DL}}$)	30 dBm
Processing time at the UE (T_P)	7 OFDM symbols
Processing time at the UE (T_{gNB})	7 OFDM symbols
5GC delay (T_{CN})	0.1 ms
DAS delay (T_{DAS})	0.05 ms
Length of the factory floor (l)	20 m [13]
Width of the factory floor (w)	20 m
Height of the factory floor (h)	4 m
Inter-machine distance (D)	5 m
Side of the machine (S)	3 m
Number of production lines (W)	4
Number of machines (M)	4/line
Inter-machine activation period τ_a	8 ms
Packet size (Z_p)	616 Bytes
Simulation time (T_S)	7 s

Table 4.2: Structure of the DNN used in the DISNETS algorithm.

	Type	Size	Max Pool	Activation
Layer 1	Conv.	(10, 4, 4)	(3, 3)	Leaky ReLu
Layer 2	Conv.	(10, 3, 3)	(3, 3)	Leaky ReLu
Latent Layer	Linear	10	·	Leaky ReLu
Output Layer	Linear	K	·	Identity
Variance decaying factor (γ)			0.9999	
Number of optimization steps (O)			100	

DISNETS parameters The configuration of the DNN used in DISNETS to compute the non-linear context representation is reported in Table 4.2. Specifically, we consider two convolutional layers of size (ξ_1, ξ_2, ξ_3), where ξ_1 is the number of channels, and ξ_2 and ξ_3 are the kernel width and height, respectively, while the Max Pool field represents the size of the max pooling window. The dimension of the Latent Layer is set to 10. The size of the Output Layer is equal to the number of feasible single actions, i.e., orthogonal channels, K. The variance decaying factor is $\gamma = 0.9999$, and number of steps between two network updates (see Algorithm 4.1, line 10) is equal to $O = 100$. DISNETS' DNN implements the *Leaky Rectified Linear Unit (Leaky ReLu)*

activation function.

Performance metrics Numerical results are given in terms of the overhead (measured as the impact of FCI transmissions on the control plane), and the E2E latency and reliability defined in Section 4.2.2, as a function of the number of UEs and the type of traffic.

Benchmarks The performance of DISNETS is compared against the following baselines:

- GBS: it implements the standard centralized 5G NR GBS [1], which requires UEs and the gNB to exchanges scheduling requests (via the PUCCH) and grants (via the PDCCH), respectively, before transmitting data. In this case resource allocation is based on the number and size of packets that the UEs have in their queues when transmitting scheduling requests via the PUCCH.
- SPS: it implements the standard centralized 5G NR SPS [6], in which the gNB allocates (part of) the resources to the UEs semi-statically over a certain time interval. This approach promotes lower latency as resources are assigned only when UEs generate packets, and without additional message exchange during transmission.
- NLTS: it implements distributed resource allocation based on the NLTS algorithm proposed in [210] and described in Section 4.2.4, thus with the assumption that UEs can transmit through one single orthogonal channel.
- RandomK: it implements distributed resource allocation in which UEs use exactly K^* orthogonal channels for transmission, and K^* is optimized through exhaustive search.

Training Convergence

First, we study the training performance of the proposed DISNETS framework considering uniformly aperiodic traffic, with $t_{min} = 2$ ms, $t_{max} = 6$ ms, and $N = 60$ UEs. Fig. 4.2 (top) plots the average and standard deviation of the training loss of the NLTS module of DISNETS (specifically, the DNN), which is used to learn the non-linear representation of the context $\phi_\omega(s)$. The loss decreases quite quickly, and becomes stable after around 0.5 s, which is an indication of the accuracy of the DISNETS implementation. In fact, as the training progresses, UEs are learning to allocate resources more accurately.

In Fig. 4.2 (center) we plot the statistics of the reward obtained by the agents as a function of the total number of interactions with the environment. We can see that, at the beginning, the reward is close to -1 given that all UEs start making random decisions in terms of the orthogonal channels to use at each SU. Then, the average reward is an increasing function of the number of steps, meaning that the UEs are learning to make more accurate allocations as the training progresses. Interestingly, we can recognize two training phases. At first the UEs are learning fast, at the rate of convergence of the DNN: in this phase, the average reward increases steeply. This is motivated by the fact that, at the beginning of the training phase, many collisions occur, which

gives the DISNETS more chances to optimize based on the relative rewards. Then, DISNETS takes more time to achieve better cooperation, and the framework optimizes more slowly.

In Fig. 4.2 (bottom) we plot the statistics of the E2E latency experienced by the UEs as a function of the packet ID. Again, it is possible to separate the two training regimes. At first, DISNETS can estimate the number of orthogonal channels to use to minimize collisions starting from a random guess, and achieves an average latency of around 1 ms. Then, DISNETS is used to further optimize resource allocation reducing the latency to around 0.7 ms, though taking more time to converge.

Overhead

As described in Section 4.2.4, DISNETS requires periodic FCI transmissions, which include the transmission outcomes (successful transmission, collision, or outage) relative to every UE on each orthogonal channel, and is used to generate the context and the reward. Therefore, the size of the FCI (in bits) can be computed as $K \cdot \log_2(N + 3)$ (see Table 4.3).

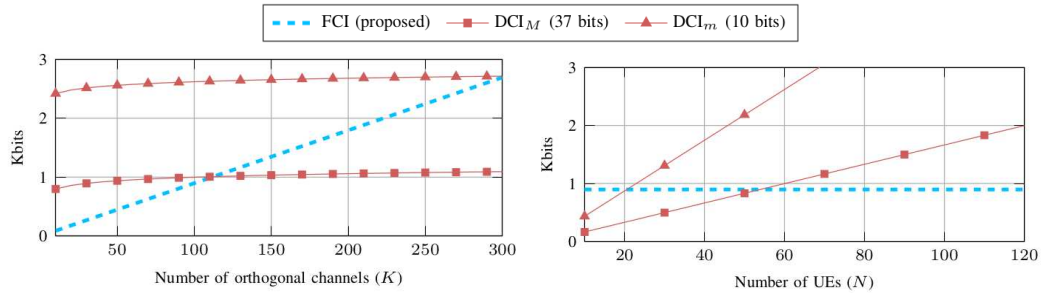
The structure of the FCI is similar to that of the 5G NR DCI signal [185], which is used to handle downlink transmissions. The size of the DCI depends on the number of orthogonal channels. Moreover, while the FCI embeds information for all the UEs, the DCI requires UE-specific transmissions, so the overall size (in bits) goes as $N_a \cdot \log_2(K)$. Furthermore, the DCI requires an additional (variable) number of bits to carry information for, e.g., carrier aggregation, Hybrid-ARQ, frequency allocation, channel access [185, Section 10.1.4]. 3GPP NR defines 10+ different DCI formats. For simplicity, in this work we consider two representative DCI formats, referred to as DCI_m and DCI_M, which require 10 and 37 additional bits, respectively, and the relative DCI size is reported in Table 4.3.

Based on the above introduction, in Fig. 4.9 we plot the size of the FCI, DCI_m, and DCI_M signals, which is directly proportional to the overhead. Specifically, we see that the size of the FCI scales linearly with the number of orthogonal channels (Fig. 4.9a) and logarithmically with the number of UEs (Fig. 4.9b), while for the DCI it is almost the opposite. As such, DISNETS achieves comparable or lower overhead than other solutions based on the DCI (e.g., 5G NR GBS) in many reasonable configurations. Notice that the results in Fig. 4.9 do not account for the additional overhead introduced to send (receive) scheduling requests (grants) in GBS, which is not required in DISNETS since resource allocation is distributed.

As another measure of overhead, in Fig. 4.10 we plot the empirical cumulative distribution function (CDF) of the number of orthogonal channels used by the UEs at each scheduling opportunity for DISNETS vs. RandomK, which is an indication of the channel occupancy. Statistics are

Table 4.3: Size (in bits) of the FCI and DCI signals, vs. the number of active UEs (N_a) and the number of orthogonal channels (K), with N fixed to 500.

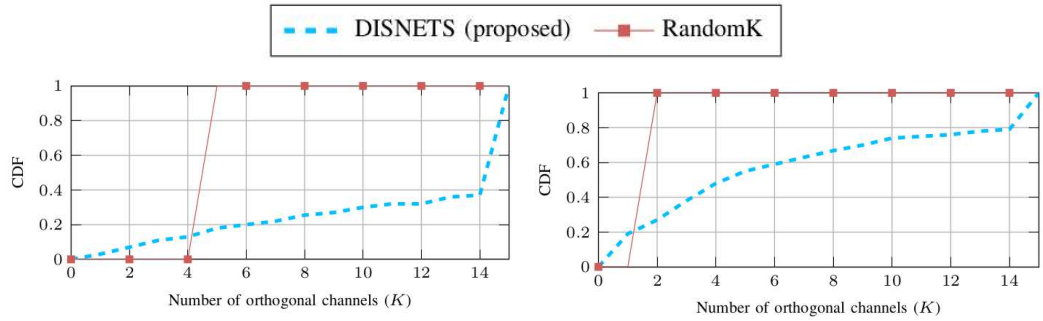
FCI	DCI	
	DCI_m	DCI_M
$K \cdot \log_2(N + 2)$	$N \cdot \log_2 K + 10$	$N \cdot \log_2 K + 37$



(a) Impact of the number of orthogonal channels. We set $N_a = 60$ and $N = 500$.

(b) Impact of the number of UEs. We set $K = 100$ and $N = 500$.

Figure 4.9: Overhead performance measured in terms of the size of the the FCI (proposed) vs. the 3GPP NR DCI, as a function of the number of orthogonal channels (top) and UEs (bottom). We consider two DCI formats, namely DCI$_m$ and DCI$_M$, which require up to 10 and 37 additional bits, respectively, for resource allocation [185].



(a) 40 UEs.

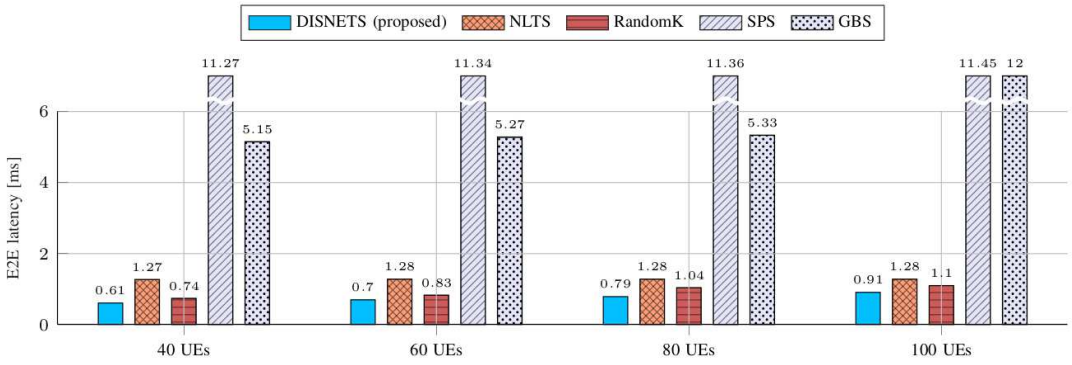
(b) 100 UEs.

Figure 4.10: Empirical CDF of the number of orthogonal channels used at each scheduling opportunity relative to the last 10 packets considering DISNETS vs. RandomK, as a function of the number of UEs.

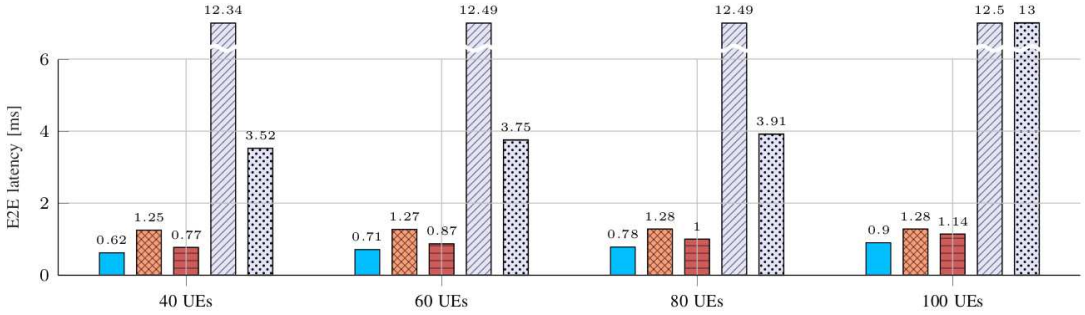
referred to the last 10 packets, i.e., after the convergence of DISNETS. We observe that RandomK uses a constant (though optimized) number of channels equal to $K^* = \{5, 2\}$ for $N = \{40, 100\}$ respectively, so K^* is a decreasing function of N. This is due to the fact that, as the number of UEs increases, the number of collisions also increases: in these conditions, the system is encouraged to reduce the number of channels to use to reduce the congestion. On the other hand, the adaptability and flexibility features of DISNETS make the number of channels to use for transmission vary significantly; as such, UEs are free to optimize the number of resources as a function of N.

Performance Evaluation

Impact of the number of UEs and the resource allocation strategy We now compare the performance of DISNETS against the RandomK, NLTS, GBS, and SPS baselines as a function



(a) Uniformly aperiodic traffic, with $t_{min} = 2$ ms and $t_{max} = 6$ ms.



(b) UE-specific aperiodic traffic.

Figure 4.11: Average E2E latency for DISNETS vs. RandomK, NLTS, SPS, and GBS, as a function of the number of UEs and the type of traffic.

of the number of UEs in the system (N). Fig. 4.11a reports the E2E latency considering uniformly aperiodic traffic. As expected, the E2E latency increases as N increases given that the network is more congested, which increases the probability of collision and re-transmissions. Also, we can see that DISNETS always outperforms all the benchmarks. In particular, centralized GBS is not able to satisfy the $L_{\text{th}} = 1$ ms requirement of URLLC due to the additional delays introduced to send (receive) scheduling requests (grants), especially when the number of UEs increases. Interestingly, SPS underperforms GBS in most configurations. In fact, SPS is designed to work well as long as the traffic is periodic [74]: in this case, SPS can pre-allocate resources based on the actual traffic periodicity by the Radio Resource Control (RRC), and does not require the UEs and the gNB to exchange additional messages. However, for aperiodic traffic as in Fig. 4.11, SPS may not be able to react to possible (unpredictable) changes in the traffic patterns and requests, with respect to how resources were originally pre-assigned, which implies that the system may operate in a sub-optimal configuration [56]. As such, unscheduled UEs will keep data packets in the queue, thus accumulating delays, at least until SPS is re-configured by another RRC interaction.

Compared to another distributed benchmark such as RandomK, DISNETS can reduce the E2E latency by up to 20%. In fact, DISNETS exploits coordination, and is designed to optimize

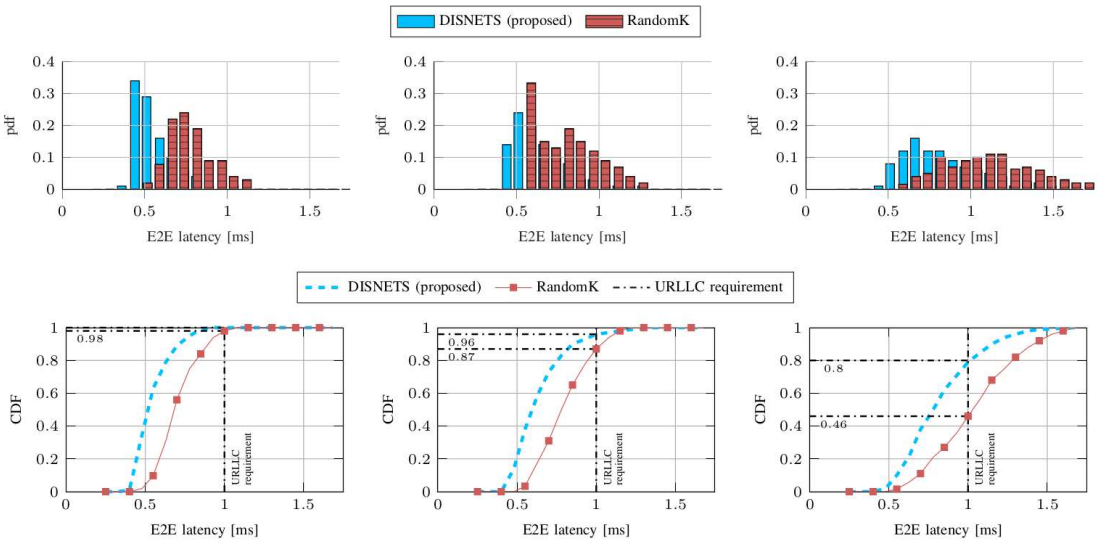
resource allocation depending on the type of traffic and the relative load of machines (for example allocating more resources to machines with more UEs). In the end, the latency for NLTS is up to 1.71× and 2× higher than DISNETS and RandomK, respectively, given that UEs can use only one orthogonal channel. While this approach brings the probability of collision to almost zero, it leaves the network underutilized. In comparison, RandomK uses K^* channels, while DISNETS can dynamically adapt the number of channels to optimize the trade-off between latency and collision.

Moreover, in Fig. 4.11b we consider the case of UE-specific aperiodic traffic. We observe that DISNETS and GBS can exploit the additional degrees of correlation introduced in the traffic to improve the latency compared to Fig. 4.11a. This is particularly true for $N \leq 80$, after which performance degrades quickly due to congestion. Still, DISNETS is the only scheme able to satisfy the $L_{th} = 1$ ms latency requirement in all configurations as it learns more from the correlation in the packet generation process. Notice that the latency for RandomK and SPS is slightly higher than in the scenario in Fig. 4.11a due to the fact that the extra packets generated in the interval $[t_{\min}, t_{\max}]$ may create more collisions. A similar observation holds for the NLTS baseline.

We recall that URLLC requires both low latency and high reliability. In Section 4.2.2 we defined reliability $\eta_t(L_{\text{th}})$ as the probability that the E2E latency associated with a packet is below a pre-defined requirement (here set to $L_{\text{th}} = 1$ ms). To capture this trend, in Fig. 4.12 we plot the Probability Density Function (PDF) (top) and CDF (bottom) of the E2E latency for DISNETS vs. RandomK (the two best solutions for resource allocation based on the previous results). We can see that the latency distributions for DISNETS are strongly shifted towards the left compared to RandomK (an indication of a smaller E2E latency), and the gap increases as N increases. For example, while for $N_a = 40$ both systems achieve comparable performance, for $N_a = 100$ (see Fig. 4.12) we have that only 46% of the UEs experience an E2E lower than $L_{\text{th}} = 1$ ms using RandomK, vs. 80% for DISNETS.

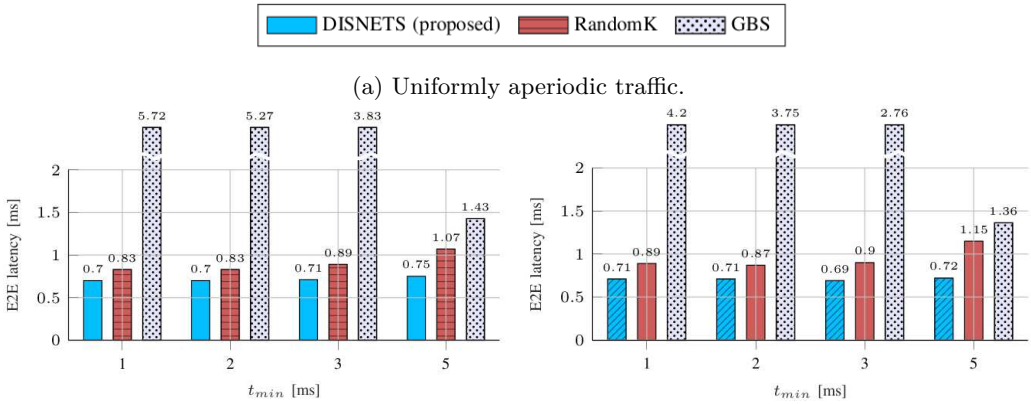
Impact of the type of traffic From the previous paragraphs we concluded that SPS and NLTS are not compatible with URLLC. So, in this set of experiments we focus on DISNETS, RandomK, and GBS as a function of the type of traffic. First, in Fig. 4.13 we change the value of t_{min}, which is inversely proportional to the traffic load, considering both uniformly aperiodic (Fig. 4.13b) and UE-specific aperiodic traffic (Fig. 4.13c). We can see that DISNETS is better than any other benchmark, and the latency is consistently below 1 ms in all configurations. Notice that for GBS the E2E latency increases as t_{min} decreases because the traffic is more intense and the system is more congested. On the contrary, for RandomK and DISNETS we have the opposite trend, i.e., the E2E latency increases as t_{min} increases. This is motivated by the fact that, as t_{min} approaches $t_{max} = 6$ ms, the traffic becomes quasi-deterministic and the UEs tend to generate packets almost simultaneously, which may increase the number of collisions. Consequently, achieving coordination becomes harder. Still, for DISNETS the latency grows by as little as 7%, from 0.7 for $t_{min} = 1$ ms to 0.75 ms for $t_{min} = 5$ ms. In turn, the performance of RandomK deteriorates significantly as t_{min} increases, and almost achieves the same performance as GBS in the long term.

Finally, in Fig. 4.14 we study the E2E latency as a function of the percentage of aperiodic UEs



(a) 40 UEs. (b) 60 UEs. (c) 100 UEs.

Figure 4.12: Empirical PDF (top) and CDF (bottom) of the E2E latency considering DISNETS vs. RandomK as a function of the number of UEs. We consider uniformly aperiodic traffic, with $t_{min} = 2$ ms and $t_{max} = 6$ ms.



(a) Uniformly aperiodic traffic.

(b) Uniformly aperiodic traffic. (c) UE-specific aperiodic traffic.

Figure 4.13: Average E2E latency for DISNETS, RandomK, and GBS, as a function of t_{min} and the type of traffic. We set $t_{max} = 6$ ms.

in the network. Specifically, for the fraction of aperiodic UEs we set $t_{min} = 2$ ms and $t_{max} = 6$ ms, whereas for the periodic UEs we set a periodicity $\tau = 2$ ms. As such, the average inter-packet interval for aperiodic UEs is equal to 4 ms vs. $\tau = 2$ ms for periodic UEs, which means that the latter generates more traffic. Again, we see that DISNETS outperforms the other benchmarks, and is therefore able to work well in both periodic and mixed, i.e., periodic and aperiodic, traffic

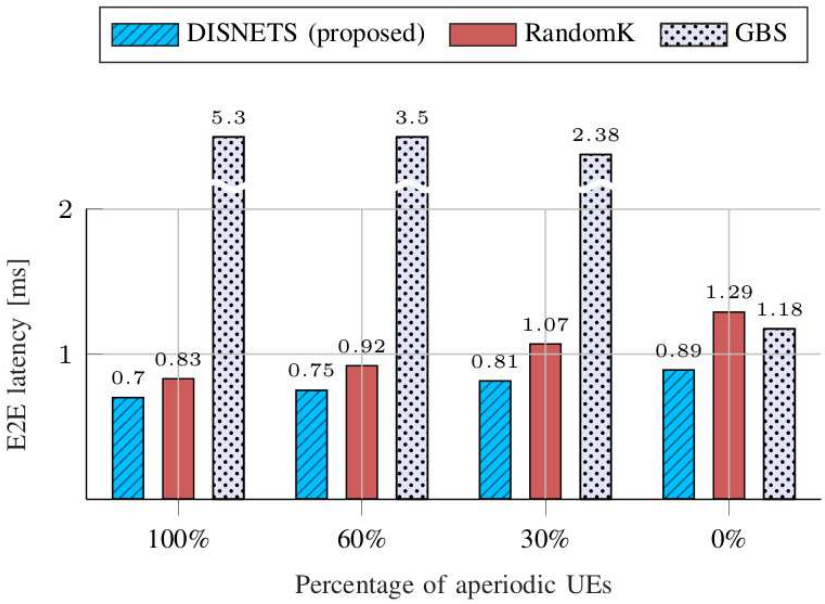


Figure 4.14: Average E2E latency for DISNETS, RandomK, and GBS, as a function of the percentage of aperiodic UEs. For aperiodic traffic we set $t_{min} = 2$ ms and $t_{max} = 6$ ms, while for periodic traffic we set $\tau = 2$ ms.

conditions. Notably, the performance of GBS decreases as the traffic becomes more aperiodic. This is expected, and we proved in [74, 56] that GBS, and SPS do not work well for unpredictable aperiodic traffic. On the other hand, as mentioned in Fig. 4.13, DISNETS and RandomK have the opposite trend, and suffer more when the traffic becomes periodic. Still, DISNETS is able to converge to good and stable results, and decrease the latency by up to 86% and 50% compared to GBS and RandomK, respectively.

4.2.6 Conclusion

In this work we shed light on the issue of enabling URLLC in IIoT networks. Specifically, we focused on the impact of resource allocation on the E2E latency. While the two main 5G NR centralized schedulers, namely GBS and SPS, have been proven to fail when considering aperiodic (unpredictable) traffic, we proposed the implementation of a new distributed scheduling framework called DISNETS, that combines DNN and LTS to allow end users to autonomously optimize their uplink transmissions, disaggregated from the network. Specifically, we described the problem in the form of a MA-CC-MAB, and described the DISNETS as our proposed solution. Specifically, DISNETS introduces new functionalities, including (i) a new control signaling scheme called FCI to train the DISNETS framework, and (ii) a new protocol procedure for autonomously selecting multiple radio resources to reduce the probability of collision. We showed via simulations that DISNETS is compatible with URLLC even for aperiodic traffic and considering IIoT-specific

correlations, and outperforms state-of-the-art centralized and decentralized benchmarks.

4.3 Distributed Reinforcement Learning for Flexible and Efficient UAV Swarm Control

4.3.1 Introduction

The high data rate achievable with modern wireless communications and the increasing computational power of embedded systems, along with the sharp price reduction of commercial Unmanned Aerial Vehicles (UAVs), have enabled the use of swarms of drones for Smart City services [115]. Thanks to their size, flexibility and flight ability, these swarms represent a new solution for a plethora of different applications, such as remote surveillance, distributed sensing, wireless coverage extension and object tracking [221].

Over the past few years, researchers have proposed several UAV-based systems [220], but achieving an efficient distributed control is a complex problem, whose solution is often task-dependent. In this context, it is important to properly define the different sub-tasks of surveillance, monitoring, mapping and tracking [64]. In this work, we assume that targets are static, but occupy random positions in the monitored area. Moving UAVs are equipped with sensors that can detect targets within a limited sensing range, and a radio interface that makes it possible to share position information and sensing data. The UAVs need to coordinate to explore the area and find the targets without colliding with each other or with obstacles.

The problem of identifying fixed targets arises in several practical situations, ranging from the generation of real-time flood maps [34] to the detailed tracking of weeds in agriculture [22], but an efficient initial exploration is of interest even for larger classes of problems, e.g., considering moving targets. One such example is wildfire monitoring in dry regions [130], which can be effective as long as the UAVs move faster than the spread of the fires.

The dynamic nature of these problems, in which actions can have long-term consequences and affect the future evolution of the environment in complex ways, makes them a natural application area for reinforcement learning (RL) techniques [233]. However, due to the curse of dimensionality, a centralized approach to the problem (i.e., using a single controller) is feasible only for very small swarms. In order to design a scalable system, multi-agent reinforcement learning (MARL) techniques need to be used, but the non-stationarity of the environment [114] complicates the system design and the agent training. This additional complexity makes MARL an open research field, and the different degrees of centralization and communication between agents make the configuration of the learning system an interesting problem to investigate.

In this work, we consider a MARL framework for exploration and surveillance. Our aim is to find a flexible machine learning (ML) strategy to explore and monitor a certain area with a swarm of UAVs that can exchange information within a certain coverage range. Performance is determined by the ability of the drones to find and reach the targets, which are located in unknown positions.

In our framework, the observations of other agents are shared through a radio channel and used to make decisions and to avoid collisions, thus encouraging cooperation. We define a Deep Q-Network (DQN) algorithm and demonstrate its efficiency with limited training, comparing it to a benchmark look-ahead heuristic and showing that our approach can better explore the environment and reach the targets faster. We also perform a transfer learning experiment, showing that agents trained on a certain map can learn to adapt to a completely new scenario much faster than restarting the training from scratch.

We adopted a general model, using a grid-world representation and making a limited number of assumptions on the nature of the task. Nonetheless, we show that our system can be implemented in several different scenarios. In particular, the map is not entirely visible to the UAVs, there are obstacles, and targets are in unknown positions (often clustered together, making clusters rarer and thus harder to find). These features make MARL highly complex, especially when considering limited communication capabilities: to the best of our knowledge, our work is the first to apply it in such challenging conditions.

Our approach to solve the problem is to model the state as a series of correlated maps, which contain different information on the environment, making the learning framework extendable to even more complicated scenarios.

The contributions of this work can be summarized as follows:

- We formulate a Networked Distributed Partially Observable Markov Decision Process (ND-POMDP) framework for swarm management in a complex environment and propose a MARL architecture to address such a problem;
- We show that the proposed system can outperform computationally heavy heuristics and transfer its knowledge to different scenarios with limited retraining;
- We analyze the effect of bigger changes in the environment, such as changing the size of the map or the number of drones, and show that transfer learning is still effective;
- We show that the system is robust to channel impairments, and can perform very well even in realistic scenarios that differ from the more abstract models used in the training phase.

A preliminary version of this analysis was also presented at ACM DroNet 2020 [249]; this version has a significantly updated system model, considering different map sizes and the presence of obstacles as well as a different MARL solution, and more extensive results on the performance of our approach. Moreover, we have added the analysis of the impact of the communication channel on the system's performance, and tested the proposed solution in a map obtained from real data.

4.3.2 Related Work

An extensive taxonomy of multi-agent solutions was presented in [53]. The general approaches adopted to solve the MARL problem can be cast into one of these four frameworks: (1) a single agent architecture that interacts with multiple copies of itself, generating emergent behaviors; (2) communication between agents of the same type with improved coordination; (3) cooperation

between agents with different specialized goals achieving coordinated behavior; and (4) modeling other agents' behaviors and planning a response [113].

The authors in [274] study the first of these four approaches and use the tabular Q-learning algorithm to guide drones to survey an unknown area, showing that even the simplest MARL algorithm can improve the overall system rewards. Similarly, in [73] and [72] the MARL framework is applied to a more complex problem in which a UAV network is adopted to provide flexible wireless communication. However, in these works the MARL algorithm is used to optimize resource allocation instead of guiding drones, so that a coordinated exploration strategy is missing.

An interesting research direction for MARL is pioneered in [92], which uses deep neural networks (DNNs) to represent and learn more complex Q-functions [175]. At first, the authors study the performance of one network trained for all agents, which then share the same parameters during the execution phase (this is also our approach). A second proposed system uses the Differentiable Inter-Agent Learning (DIAL) framework, in which agents learn meaningful real-valued messages to be exchanged in order to improve cooperation: this allows for faster training, but the model is limited to a very small number of agents.

Other works use RL in the practical scenarios discussed above: in [34], the authors adopt a MARL approach to control a flood-finding swarm of UAVs. However, the model only considers a swarm with a fixed number of drones, and the experimental results are not compared to state-of-the-art heuristics. In [22], a reinforced random walk model is exploited to map weeds in an agricultural setting, taking noisy acquisition into account and solving the issue with collective observations. Random walks are then biased based on the positions of the already discovered targets, which have to be properly mapped, along with the distances from other drones in the network. In this case, the authors considered swarms of variable sizes, but the random walk needs to be manually tuned for each setting. Another recent study [130] considers wildfire spread monitoring, checking how the fire evolves and spreads in the map from a known starting point. The authors define the problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [182] and carry out several experiments, as well as comparisons against a greedy heuristic (similar to the look-head method we studied in this work). A target-tracking application for disaster scenarios, with a model similar to our own but applied to a single drone, is described in [264]. Finally, [117] considers a MARL system with realistic communication, where a swarm of drones needs to get data from an Internet of Things (IoT) sensor network. This is a much simpler problem, as the position of the targets is known in advance, and the MARL framework only needs to optimize the trajectories.

The MARL approaches can also fit models in which UAV connectivity is important: in [57], a framework including RL and game theory is used to plan the path of two drones that need to save energy and minimize the interference to the ground network while maintaining a cellular connection. Furthermore, in [161] the authors design a centralized RL system to maximize coverage for a swarm of aerial base stations serving mobile users on the ground. A similar approach is taken in [164], which redefines the problem in terms of Quality of Experience (QoE) maximization for the users. For a fuller communication-oriented perspective on the use of RL for UAV networks, we refer the reader to [118].

Symbol	Description	Symbol	Description
$\mathcal{M}$	Coordinate set	$\mathcal{O}$	Observation space of the system ND-POMDP
M	Map grid size	$\mathbf{\Phi}$	Matrix of cell values
K	Number of targets	$\mathbf{X}$	Matrix of UAV positions
$\mathbf{z}_k$	Coordinates of the k-th target	$\mathbf{\Omega}$	Matrix of obstacle positions
σ	Standard dev. of the target Gaussian functions	$\hat{\mathbf{\Phi}}$	Observed cell value matrix
$\phi(\cdot)$	Cell value function	$\hat{\mathbf{\Omega}}$	Observed obstacle position matrix
$\mathcal{U}$	Set of UAVs	$\mathbf{X}_u$	Observed UAV position matrix for u
U	Number of UAVs (cardinality of $\mathcal{U}$)	F	Observation window size (in number of cells)
d_{sparse}	Minimum target distance in the sparse scenario	ψ	Penalty for collisions
$\omega(\cdot)$	Obstacle location function	θ	Penalty for moving to forbidden areas
η	Fraction of the map occupied by obstacles	ρ	Obstacle value
ζ	Field of View of each UAV	$\nu_u(\mathbf{x}_u, \mathbf{a}_u)$	Invalid move indicator function for UAV u
$h_{\min}$	Minimum obstacle size (in number of cells)	$\chi_u(\mathbf{X}, \mathbf{A})$	Collision indicator function for UAV u
$h_{\max}$	Maximum obstacle size (in number of cells)	$r_u(s, \mathbf{a})$	Reward for UAV u
$\boldsymbol{\ell}_i$	Lower left corner coordinates of the i-th obstacle	π	Observation-action policy
$\mathcal{H}_i$	Set of cells occupied by the i-th obstacle	$R_{u,t}(\pi)$	Long-term reward for u using policy π
$\mathbf{h}_i$	Size of the i-th obstacle	γ	Exponential discount factor
N	Episode duration (steps)	e_t	Experience sample
$\mathcal{S}$	State space of the system ND-POMDP	α	Learning rate
$\mathcal{V}(\mathbf{s})$	Valid move space for state s	B_{size}	Size of a learning batch
$\mathcal{A}$	Action set	$Q(o_u, a_u)$	Q-value estimate of R
$\mathbf{a}_u$	Action for UAV u	n_q	Model update period steps)

Table 4.4: Notation definitions.

These works have similar objectives to our own, but either go back to the single-agent setting or have restrictive assumptions: as an example, [130] considers well-known fire patterns, which can be extensively learned, with a known starting point. In our case, the initial positions of the targets and of the UAVs are not the same across different episodes, making the model more general and complicating the learning task. Furthermore, unlike previous efforts in the literature, we exploit the transfer learning paradigm, showing how our model can easily adapt to scenarios with obstacles, realistic maps, and different swarm sizes. To the best of our knowledge, our work presents the most complex environment to date, in which a single architecture can deal with different map and swarm sizes, different numbers of targets to track, and the presence of obstacles.

4.3.3 System Model

In the following, we first present the environment in which the UAVs operate. We give a full list of the notation used in Table 4.4 as a reference to the reader.

Environment

The system environment consists of a square grid of size $M \times M$. Each cell of the grid (we will refer to a cell or a location interchangeably in the following) is identified by its coordinates $\mathbf{m} \in \mathcal{M}$, where $\mathcal{M} = \mathcal{X} \times \mathcal{Y}$, and $\mathcal{X} = \mathcal{Y} = \{0, ..., M-1\}$. We place a set of K targets on the map, which represent the objectives of the UAV surveillance application. The position of the k-th target is denoted as $\mathbf{z}_k = (x_k, y_k)$.

We then generate a set of K bivariate Gaussian functions over the grid, which represent the visibility of a target to the UAVs, with the same covariance matrix $\Sigma = \left(\begin{smallmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{smallmatrix}\right)$. The mean

$\mathbf{z}_k = (z_{k,1}, z_{k,2})$ corresponds to the coordinates of the target. Note that the Gaussian functions do not represent actual distributions, but rather the full view of the UAVs, which can see a target from afar. The value of σ can be interpreted as the distance at which a target can be identified, as larger values of σ mean that the target is visible from further away.

Each cell can then be associated with a weight $\phi(\mathbf{m})$, which represents the *value* of the location, which increases with the proximity to a target, and is given by the maximum of the Gaussian functions in that point, normalized in such a way that the target locations have values equal to 1:

$$\phi(\mathbf{m}) = \max_{k \in \{0,\dots,K-1\}} e^{-\frac{1}{2}\left((\mathbf{m}-\mathbf{z}_k)^{\mathbf{T}}\Sigma(\mathbf{m}-\mathbf{z}_k)\right)}. \tag{4.12}$$

If $\phi(\mathbf{m})$ is smaller than 0.01, it is set to 0, as the UAVs cannot see any target from that location. Under these conditions, the most valuable cells coincide with the center of each Gaussian function, which represents one of the targets in the considered scenario. While the environment is static, the UAVs move within the map with the aim of positioning themselves over the targets as fast as possible. We denote the set of UAVs by $\mathcal{U}$, and by U its cardinality.

In this work, we consider two different distributions for the targets, named *sparse* and *cluster*, which are characterized by different correlations among the target positions. In both cases, the first target is randomly placed on the grid following a 2D uniform distribution: $\mathbf{z}_0$ can take any value in $\mathcal{M}$ with equal probability. The other targets are then placed sequentially, according to the following rules. In the sparse scenario, the position $\mathbf{z}_i$ of the i-th target is randomly chosen in the set $\mathcal{M}_i^{\text{sparse}} = \{\mathbf{m} \in \mathcal{M} : ||\mathbf{m} - \mathbf{z}_j||_2 > d_{\text{sparse}},\ \forall j < i\}$, with probability mass distribution

$$P_{\text{sparse}}(\mathbf{z}_i = \mathbf{m}) = \frac{||\mathbf{m} - \mathbf{z}_0||_2}{\kappa_i^{\text{sparse}}}. \tag{4.13}$$

where $\kappa_i^{\text{sparse}} = \sum_{\mathbf{m} \in \mathcal{M}_i^{\text{sparse}}} ||\mathbf{m} - \mathbf{z}_0||_2$ is a normalization factor. Hence, the other targets tend to be distributed far from the first, with a minimum distance d_{sparse} between each other.

In the cluster scenario, instead, the i-th target can take any position in the set $\mathcal{M}_i^{\text{cluster}} = \{\mathbf{m} \in \mathcal{M} : ||\mathbf{m} - \mathbf{z}_j||_2 > 1,\ \forall j < i\}$ with probability mass distribution

$$P_{\text{cluster}}(\mathbf{z}_i = \mathbf{m}) = \frac{1}{(1 + ||\mathbf{m} - \mathbf{z}_0||_2)\kappa_i^{\text{cluster}}}. \tag{4.14}$$

where $\kappa_i^{\text{cluster}} = \sum_{\mathbf{m} \in \mathcal{M}_i^{\text{cluster}}} \frac{1}{(1+||\mathbf{m}-\mathbf{z}_0||_2)}$ is the normalization factor. In this case, the targets tend to cluster around the first one, but cannot occupy adjacent cells, since the minimum distance must be greater than 1.

An example of the two target placements is shown in Fig. 4.15. These two distributions represent two plausible configurations of targets in tracking applications: in wildlife monitoring, some species of animals might tend to herd together, while more territorial ones will have a sparser distribution on the map. The same goes for a battlefield scenario, in which groups of soldiers might act together as a tight formation, while guerrilla-style fighting will involve a much sparser distribution of forces.

In a more complex version of the scenario, the map does not just have targets that the UAVs need to find and reach, but *obstacles* as well: in an urban scenario, these might be tall buildings

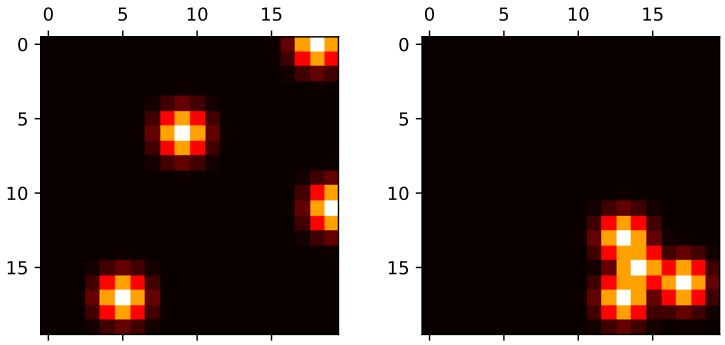


Figure 4.15: Two examples of the sparse (left) and cluster (right) target distributions.

or designated no-fly zones, while in a natural scenario they might correspond to natural obstacles such as boulders or tall trees. We define a function $\omega(\mathbf{m})$, which is equal to 1 if the cell corresponds to an obstacle and 0 otherwise. Then, we denote by η the portion of the map occupied by obstacles:

$$\eta = \sum_{\mathbf{m} \in \mathcal{M}} \frac{\omega(\mathbf{m})}{M^2}. \tag{4.15}$$

Cells inside an obstacle are considered impassable, like the map borders, and the UAVs that try to move on an obstacle will remain in the same cell.

For the training of our algorithm, we assumed that obstacles are rectangular and randomly scattered in the area. The i-th obstacle is determined by its dimensions $\mathbf{h}_i$ and by the position of its lower left corner $\boldsymbol{\ell}_i$. We formally define the obstacle as the set $\mathcal{H}_i$:

$$\begin{aligned} \mathcal{H}_i = \big\{ & \mathbf{m} = (m_1, m_2) \in \mathcal{M} : \\ & m_1 \in \{\ell_{i,1}, \dots, \ell_{i,1} + h_{i,1} - 1\}, \\ & m_2 \in \{\ell_{i,2}, \dots, \ell_{i,2} + h_{i,2} - 1\}\big\}. \end{aligned} \tag{4.16}$$

Obstacles are generated sequentially, like the targets, and for each obstacle i the dimensions $\mathbf{h}_i$ are drawn uniformly from the set $\{h_{\min}, h_{\max}\} \times \{h_{\min}, h_{\max}\}$. The lower left corner position $\boldsymbol{\ell}_i$ is then drawn from a uniform distribution in the set $\mathcal{M}_i^{\text{obs}}$, the subset of the map defined as:

$$\begin{aligned} \mathcal{M}_i^{\text{obs}} = \big\{ & \boldsymbol{\ell} \in \mathcal{M} : \mathcal{H}_i \subset \mathcal{M}, ||\mathbf{n}, \mathbf{z}_k||_2 > 1, \\ & \forall \mathbf{n} \in \mathcal{H}_i, k \in \{0, \dots, K-1\}, \\ & d(\mathcal{H}_i, \mathcal{H}_j) \geq 2, \forall j < i\big\}, \end{aligned} \tag{4.17}$$

where $d(\mathcal{H}_i, \mathcal{H}_j) = \min_{\mathbf{m}_i \in \mathcal{H}_i, \mathbf{m}_j \in \mathcal{H}_j} ||\mathbf{m}_i - \mathbf{m}_j||_2$ is the distance between the obstacles i and j. The three constraints force the obstacle to be entirely inside the map, not to be directly adjacent to any of the targets, and not to touch other obstacles. The choice of these constraints was motivated by the necessity to guarantee the existence of a clear path to the targets from any point in the

map.

We consider multiple *episodes* of N steps: in each episode, the targets, UAVs, and obstacles are redistributed in the map, and the swarm must locate the targets in as few steps as possible. We consider discrete time slots, so that each drone can move by a single cell at each time step. Furthermore, we assume that a UAV has a limited Field of View (FoV), i.e., it can only know the value of the cells within a radius ζ. This framework allows us to represent many different applications and scenarios by changing the size of the grid, the number of drones, targets and obstacles, the FoV range ζ and the target visibility parameter σ. It can also be easily extended to dynamic targets.

At the beginning of each episode, each UAV only knows the values of the cells within the swarm's FoV. The drones assume that all unexplored points of the map are associated with the maximum $\phi(\mathbf{m})$. Then, each UAV moves independently at each time step n: as the swarm explores the environment, each drone discovers the values of the map locations that it has covered, and updates its information according to $\phi(\mathbf{m})$. We highlight that the knowledge about the map is instantly shared, which means that each drone receives the observations that all the other drones have acquired. This is always true during training, whereas in some testing episodes we also experiment the scenarios in which unreliable communications affect the shared messages. The objective of the swarm for each episode is to position each of its UAVs above a target as quickly as possible.

Communication model

We consider the swarm to only have partial observations: as the size of the map might be too large for the swarm to effectively coordinate over it, we consider each UAV to have up-to-date knowledge only inside the $F \times F$ square with it at the center, with $F \leq M$. If the distance between the UAV and the edge of the map is lower than F, the square will consider the edge of the map as the edge of the visible region, and the UAV will no longer be at its center, in order to avoid modeling the area outside the map. This assumption allows us to model communication constraints in the problem, as UAVs need to share the observed parts of the map with the other components of the swarm; however, F should not be confused with the FoV ζ, as the former represents the size of the portion of the map that each UAV considers when deciding its next action, while the latter represents the size of the portion of the map that the UAV can sense directly at each moment. In our case, we always have $F \geq \zeta$.

ND-POMDP formulation

The described scenario is modeled as an ND-POMDP [180], i.e., a Markov decision process (MDP) where the system state in not directly observable and is influenced by the actions of multiple agents, whose behavior is not centrally coordinated. Indeed, the swarm only has limited knowledge of the map, and the UAVs can take actions independently and have independent rewards. We observe that ND-POMDP is a particular class of Decentralized POMPD (Dec-POMDP) for which not all agents interact with each other [143]. Convergence to the optimal solution for this kind of

problem has been proven for classical reinforcement methods [275], although not for deep models: as most works in the literature, we will use a benchmark to evaluate the performance of our scheme. Formally, an ND-POMDP is identified by a 5-tuple, composed of a state space $\mathcal{S}$, an agent space $\mathcal{U}$, a joint action space $\mathcal{A}$, an observation space $\mathcal{O}$, and a reward map $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^U$, where $U = |\mathcal{U}|$.

The complete system state s is given by five matrices: a matrix for the current position of the UAVs, one matrix each for the map of the already discovered targets and obstacles, and one matrix each for the full map of targets and obstacles. The positions of the UAVs are contained in the $2 \times U$ matrix $\mathbf{X}$, while the features of the map are represented by the two $M \times M$ matrices $\mathbf{\Phi}$ and $\mathbf{\Omega}$, which contain the value $\phi(\mathbf{m})$ of each cell and the function $\omega(\mathbf{m})$ representing the location of the obstacles. Clearly, the maps with the full view of targets and obstacles are not initially known by the UAVs, which will then need to explore the area.

Furthermore, the UAVs do not know the features of cells that have not been explored: the observed features of the map are contained in the $F \times F$ observed value matrix $\hat{\mathbf{\Phi}}$, whose elements are equal to $\phi(\mathbf{m})$ if the cell has been explored and 1 otherwise, and the $F \times F$ observed obstacle matrix $\hat{\mathbf{\Omega}}$, whose elements are equal to $\omega(\mathbf{m})$ if the cell has been explored and 0 otherwise. The observation $o_u \in \mathcal{O}$ that is available to drone u is then given by $\mathbf{X}_u$, $\hat{\mathbf{\Phi}}_u$, and $\hat{\mathbf{\Omega}}_u$, defined as the $F \times F$ subsets of $\mathbf{X}$, $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Omega}}$ centered in $\mathbf{x}_u$.

In our case, each UAV can either stay over the same cell or move to one of the four adjacent cells. However, obstacles are impassable in our environment definition, and the UAVs cannot move outside the map, so UAVs will simply stand in place if they attempt an action that violates the constraints. We define the action space $\mathcal{A} = \{(0,0), (0,1), (1,0), (0,-1), (-1,0)\}^U$. An action for the swarm is then a vector $\mathbf{a} \in \mathcal{A}$, which contains the individual UAVs' actions, denoted as $\mathbf{a}_u$ for drone u. We first define function $\nu(\mathbf{x}_u, \mathbf{a}_u)$, which is 1 if the action is valid, i.e., it does not lead the UAV to fly outside the map or into an obstacle, and zero otherwise:

$$\nu(\mathbf{x}_u, \mathbf{a}_u) = \begin{cases} 1, & \text{if } \mathbf{x}_u + \mathbf{a}_u \in \mathcal{M} \wedge \omega(\mathbf{x}_u + \mathbf{a}_u) = 0; \\ 0, & \text{otherwise.} \end{cases} \tag{4.18}$$

The position of each drone is then updated in the following way:

$$\mathbf{x}_u(t+1) = \mathbf{x}_u(t) + \mathbf{a}_u(t)\nu(\mathbf{x}_u(t), \mathbf{a}_u(t)). \tag{4.19}$$

Fig. 4.16 shows an example of the system state at the beginning and in an advanced stage of an episode, with two drones and four targets located in a 20×20 map with no obstacles (in this case, we set $F = M = 20$). In particular, the drones' positions are shown on the left (in yellow), the observed value map is in the center, and the real value map is on the right. In the figure, darker cells are associated with lower values and brighter cells are associated with higher values. In the figure, if the communication range equals or exceeds the map side, i.e., $F \geq M$, the observed state o for all UAVs would correspond to the maps on the left and in the center. On the contrary, if $F < M$, the observation for each UAV would include a different portion of the map. It is easy to

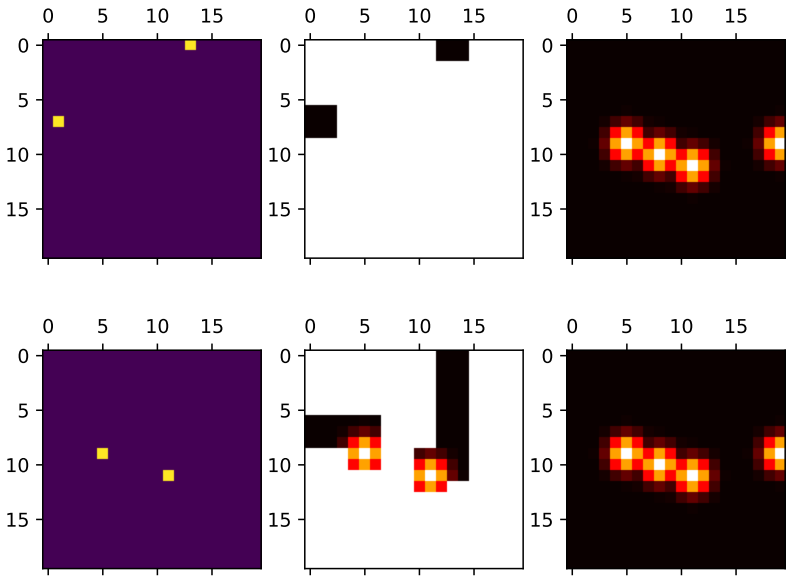


Figure 4.16: Drone positions (left), known map (center), real map (right). Beginning (above) and end (below) of an episode.

see how the swarm gains knowledge during the episode, as the drones explore the map and look for targets. In this case, the UAVs found two targets relatively quickly, and a significant portion of the grid remained unexplored.

We give reward 1 to a UAV if it is directly above a target, reward $-\theta$ if it tries to go outside the map or to position itself over an obstacle, reward $-\psi$ if it is in the same cell as another drone, and reward 0 in any other case. The UAVs will quickly learn to avoid actions that would take them outside the map or make them crash into obstacles, so the exact value of θ does not affect the final performance, but the value of ψ affects the distance that the drones try to keep from each other: if ψ is low, the drones will get close to each other if the targets are very close. Naturally, if there is a collision risk when the drones are in the same cell, the value of ψ should be high. The reward depends on $\mathbf{X}$, as well as on the action vector $\mathbf{a}$.

Indicating with $\mathbf{x}_u$ and $\mathbf{a}_u$ the position and action of drone u, we now define the collision variable $\chi_u(\mathbf{X}, \mathbf{A})$ as

$$\chi_u(\mathbf{X}, \mathbf{A}) = \max_{v \in (\mathcal{U} \setminus u)} \delta\big(\mathbf{x}_u + \mathbf{a}_u(t)\nu(\mathbf{x}_u, \mathbf{a}_u) - \mathbf{x}_v - \mathbf{a}_v(t)\nu(\mathbf{x}_v, \mathbf{a}_v)\big). \tag{4.20}$$

where $\delta(\mathbf{x})$ denotes a function that takes value 1 if the vector $\mathbf{x} = 0$, and zero otherwise. In short, $\chi_u(\mathbf{X}, \mathbf{A})$ has value 1 if one or more drones move to the same cell as drone u, and 0 otherwise. The collision variable depends on the moves of other agents, so the problem is distributed. The reward function for UAV u in state s if the swarm takes the joint action vector $\mathbf{A}$, denoted as

$r_u(s, \mathbf{A})$, is given by:

$$\begin{aligned} r_u(s, \mathbf{A}) = &- \theta(1 - \nu(\mathbf{x}_u, \mathbf{a}_u)) - \psi\chi_u(\mathbf{X}, \mathbf{A}) \\ &+ (1 - \chi_u(\mathbf{X}, \mathbf{A})) \sum_{k=0}^{K-1} \delta(\mathbf{x}_u + \mathbf{a}_u - \mathbf{z}_k). \end{aligned} \tag{4.21}$$

In our model, the state transitions and the system observations are both deterministic; therefore, both the state evolution and the observation are not affected by random events, but only by the agents' decisions. We define a policy $\pi(\mathbf{a}_u|o_u)$ as the conditioned probability for user u to take action $\mathbf{a}_u$ given an observation $o_u \in \mathcal{O}$. Under these assumptions, the goal of each drone u is to find the policy π^* that maximizes the cumulative expected future discounted reward $R_u(\pi) = \mathbb{E}\left[\sum_{\tau=0}^{+\infty} \gamma^\tau r_{u,\tau}|o_u, \pi\right]$, where $\gamma \in [0, 1)$ is a discount factor.

Distributed Deep Q-Learning

In this subsection, we will describe our Distributed Deep Q-Learning (DDQL) approach to solve the problem defined above. For the sake of readability, in the following we omit the u subscript to indicate the agent whenever possible. Each agent leverages a DQN, i.e., a neural network (NN) that takes as input the last observation o_t and returns the Q-values of the possible actions that can be taken, i.e., $Q(o_t, \mathbf{a})$, $\forall \mathbf{a} \in \mathcal{A}$. In Q-learning, the function $Q(o, \mathbf{a})$ is an estimate of the expected long-term reward R that will be achieved by choosing $\mathbf{a}$ as the next action and then following the learned policy. In our case, we maintain a single DQN during the training phase, whose values are shared by all the agents. In this work, we follow the approach from [175] and leverage a *replay memory* to store the agent experience $e_t = (o_t, a_t, r_t, o_{t+1})$. Whenever the agent carries out a training step, a batch of B_{size} elements is picked from the replay memory, allowing to separate the algorithm training from the experience acquisition. The replay memory is shared between the agents during a training phase, and a new batch is used to train the agent at every step. We highlight that, in our system, all agents are the same (single DQN), and they need to generalize the problem from a limited number of states. As it would be impossible for a single UAV to experience even just a non-negligible fraction of possible states in the training, shared replay is a critical factor in the network's generalization ability. In particular, the experience replay is extremely valuable since it allows the system to improve the variety of the training samples by getting experience from the states seen by different agents. In other scenarios, it may not be convenient to exploit a shared memory, especially when the agents have to learn different tasks.

Following the DQN example from [175], we exploit the *double Q-learning* technique to remove biases from the Q-value estimation and speed up the algorithm's convergence [245]. This means that, during the training, we maintain a *target network*, whose output $Q_t(o, a)$ is used to evaluate actions, and an *update network*, whose output $Q_u(o, a)$ is used to select the policy. In particular, the bootstrap Q-value is computed as

$$Q^{\text{new}}(o_t, a_t) = r_t + \gamma \max_a Q_t(o_{t+1}, a). \tag{4.22}$$

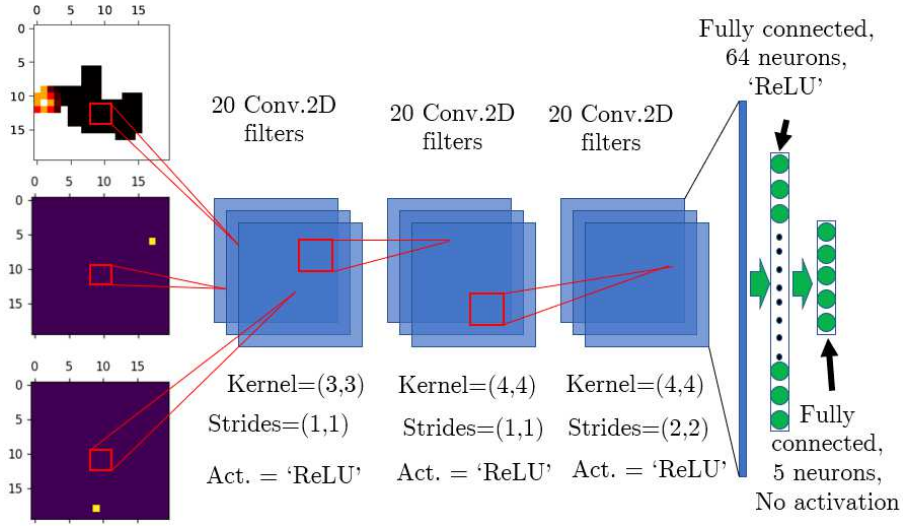


Figure 4.17: Architecture of the DQN.

The value $Q^{\text{new}}(o_t, a_t)$ is then used to perform backpropagation on the update network with a learning rate set automatically by the Rectified Adam (RAdam) optimizer [162], and every n_q training steps the update network parameters are copied to the target network.

In our model, the observed state of the system for each agent can be represented by four $F \times F$ matrices, representing the agent position, the locations of the other agents, the value of explored cells, and the position of known obstacles. To simplify the state space, we consider matrices $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Omega}}$ jointly, by feeding the NN with the matrix $\hat{\mathbf{\Phi}} - \rho\hat{\mathbf{\Omega}}$, where ρ is a scalar parameter used to facilitate learning. Therefore, our system approximates the function $Q(o, a)$ by a Convolutional Neural Network (CNN), whose architecture is described in Fig. 4.17. In particular, we consider a CNN exploiting three convolutional layers followed by two fully-connected layers. The dimension of the last layer is identical to the number of actions, so that each output element can be associated to a different action $a \in A$.

Hence, each agent provides training samples for the shared replay memory, which are then used in (4.22), so that the CNN output can converge to the Q-values $Q(o, a)$, $\forall\ a \in A$. We implement the well-known ε-greedy and softmax policies to allow the agents to explore the action space during the training phase, which is carried out by simulating a sequence of episodes.

Computational complexity

We now discuss the computational complexity to perform one inference procedure with the neural network. We first analyze the complexity of fully-connected layers. We denote by N_k the number of neurons in the general k-th layer. To go from layer i to layer $i + 1$, we need to compute the value of N_{i+1} nodes, each of which takes N_i multiplications followed by N_i additions and one non linear function, thus involving $N_{i+1}(2N_i + 1)$ operations.

We can then compute the complexity of one convolutional layer, as done in [110], when neither

batch normalization nor pooling layers are present. We denote with (I_w, I_h, I_d) the shape of the input block. At layer i, we then have K_i filters with kernels dimension (W_i, H_i), stride S_i (we use the same value along the two axes), and padding P_i. The shape of the resulting output block will be $(\frac{I_w+2P_i-W_i}{S_i}+1, \frac{I_h+2P_i-H_i}{S_i}+1, K_i)$. The computation of each block's neuron here involves $W_i \times H_i \times I_d$ multiplications followed by the same number of additions (sum all elements plus the bias) and one non-linearity. The total number of calculations is then $(\frac{I_w+2P_i-W_i}{S_i}+1) \times (\frac{I_h+2P_i-H_i}{S_i}+1) \times K_i \times (2W_iH_iI_d+1)$.

If we consider the specific architecture of our NN reported in Fig. 4.17, the actual number of basic computations (multiplications, additions and non-linearities) are, respectively, 440 000, 3 704 980 and 628 180 for the three convolutional layers. The following fully-connected layers require 125 504 and 645 computations, thus the total number of operations for one decision is 4 899 309.

This computational complexity allows UAVs to take decisions in real time, as even embedded processors can deal with much more complex architectures in less than 100 ms [45]. As the physical speed of the UAVs and the much more complex vision algorithms required to identify targets are the main limiting factors for the swarm, the ND-POMDP will be performed at a relatively slow pace, with timesteps in the order of several seconds.

4.3.4 Simulation settings

In this section, we describe the simulations by which we evaluated the performance of the designed system. All the results are derived through a Monte Carlo approach, where multiple independent simulations are carried out to obtain reliable statistical data. In particular, the algorithms' training is executed by carrying out a total of N_e episodes for each studied scenario (sparse or cluster), where each episode is given by N_s^t steps. Training episodes are far longer than test episodes, which have length N_s^p, since the agents need to explore the map fully.

Before training, we initialize the replay memory by executing $N_e^m = 1000$ episodes of N_s^t steps each, to allow agents to immediately start the learning procedure. If the episodes are too long, a lot of samples in which large portions of the map are already explored are added to the memory replay, and the agents will not learn properly how to move at the beginning of the episode, when the map is not explored. On the other hand, short episodes have the opposite problem, as the UAVs never learn to behave in the final parts of the episodes. A prioritized memory replay can solve this problem, but requires additional parameters. We then opted for adapting the episode length in the training phase. The even training episodes have 50 steps each, while the odd episodes have 150 steps. This alternating size prevents the replay memory from being too skewed towards situations in which the map is almost completely explored or unexplored.

Moreover, we apply transfer learning to allow the agents trained in the sparse environment to quickly adapt to the cluster scenarios (or vice-versa); to this goal, additional N_t training episodes are carried out. Finally, the performance of the proposed strategy is tested in a total of $N_p = 500$ episodes for the DDQL system. The exploration rate ε follows 2 different approaches, namely, ε-greedy and softmax. In the former, a random action is chosen with probability ε, while the best

Parameter	Value	Description
M	{20, 24, 30, 40, 50}	Map size
F	20	Observed map size
U	$\{2, 3\}$	Number of UAVs
K	4	Number of targets
σ^2	1	Targets variance
ζ	3	Field of View
η	{0,0.1}	Obstacle frequency
d_{sparse}	8	Minimum target distance (sparse scenario)
θ	1	Obstacle/outside penalty
ψ	0.8	Collision penalty
ρ	0.2	Obstacle value
γ	0.9	Discount factor
α	Chosen by RAdam	Learning rate
N_e	$\{250, 750, 1000, 3000\}$	Training episodes
N_s^t	$\{50, 150\}$	Steps per training episode
N_s^p	40	Steps per test episode
N_t	$\{125, 250, 375, 750\}$	Transfer learning episodes
N_p	100 (LA), 500 (DDQL)	Test episodes
P_{tx}	20 dBm	Communication power
N_0	-76 dBm	Noise floor
h	40 m	UAV height
R_c	2/3	Coding rate

Table 4.5: Simulation settings.

action, i.e., the action with the highest Q-value, is chosen with probability 1-ε. The value of ε decreases to 0 at the end of the training, since no more exploration is needed. In the latter, at each time step the probability of each action p_i is computed as the output of a softmax density function taking the Q-values as input. In this case, the temperature T decreases during the training, reducing the randomness during the selection of the actions:

$$p_i = \frac{e^{\frac{q_i}{T}}}{\sum_{j=1}^{A} e^{\frac{q_j}{T}}} \ , \tag{4.23}$$

where $A = 5$ is the number of actions that each drone can take. The training and testing processes are independently performed 5 times to verify the robustness of the DDQL scheme. The complete simulation settings are reported in Tab 4.5.

To assess the performance of our DDQL scheme, we compare it with a heuristic strategy inspired by Model Predictive Control (MPC), by which drones can explore the map and reach the targets. Such a strategy is named *look-ahead* and is used as a benchmark for our analysis. The look-ahead strategy tries all possible combinations of future actions and looks at the possible future rewards, as its name suggests. In order to define it, we first define the look-ahead reward $r_u^{(\ell)}(\mathbf{X}, \mathbf{a})$ as:

$$r_u^{(\ell)}(\mathbf{X}, \mathbf{a}) = \begin{cases} \frac{\hat{\phi}(\mathbf{x}_u + \mathbf{a}_u)}{\xi(\mathbf{x}_u + \mathbf{a}_u)} & \text{if } \nu(\mathbf{x}_u, \mathbf{a}_u) = 1; \\ -\infty & \text{otherwise,} \end{cases} \tag{4.24}$$

where $\xi(\mathbf{x})$ is the number of UAVs located in $\mathbf{x}$. The look-ahead strategy never goes outside

the map or on obstacles. To decide its next action, each drone u tries to maximize its expected cumulative reward over the following n_ℓ steps, assuming that none of the other drones move. In practice, the look-ahead strategy makes each drone select the action $\mathbf{a}^*$ that maximizes

$$\max_{\mathbb{A}\in\tilde{\mathcal{A}}^{n_\ell}} \sum_{i=0}^{n_\ell-1} r_u^{(\ell)} \left(\mathbf{X} + \sum_{j=0}^{i-1} \mathbf{A}^j, \mathbf{a}^i \right), \tag{4.25}$$

where $\tilde{\mathcal{A}}^{n_\ell}$ is the set of ordered sequences $\mathbb{A}$ of action vectors $\mathbb{A}^0$, $\mathbb{A}^1$, ..., $\mathbb{A}^{n_\ell-1}$, so that $\hat{\mathbf{a}}_u^0 = a^*$ and $\mathbf{a}_v^i = (0,0)$, $\forall\, i \in \{0,...,n-1\}, v \neq u$, i.e., the set of possible move sequences of u while the other UAVs are static. If several action sequences have the same expected reward, the look-ahead strategy will choose one of them randomly. At the beginning of an episode, each drone u assumes that all the map values $\phi(\mathbf{m})$ outside its FoV are equal to 1; therefore, look-ahead forces u to continuously explore the map. However, as soon as it finds a target, u will hover over the target center. The target is then eliminated from the other agents' value maps, as it is already covered by a UAV. We highlight that the performance of look-ahead mainly depends on the n_ℓ parameter: as it increases, drones can make more foresighted decisions, but at a greater computational cost. In addition, the number of targets in the map also plays a key role in determining the computational performance: when more targets are present, we have to check whether other agents are on a target more often, in order to remove it from the map of available targets. As the look-ahead strategy is computationally expensive, N_p for it was set to 100.

Finally, we also consider a scenario with a realistic communication model, in which the broadcast messages sent by each UAV at every step might be lost due to the wireless channel impairments. We used the path loss and shadowing model from [163], based on actual measurements from air-to-air communications, and considering a Rayleigh fading model with an error correction code with rate 2/3. As the simulation results will show, the physical size of the cells in the map is a critical parameter when UAVs communicate directly with each other (and not through the network infrastructure on the ground). In particular, increasing the size of the cells will impair the performance because of communication range issues: the model has an error probability of 50% at approximately 110 m, corresponding to 11 cells if a cell side is 10 m and 5 cells if the side is 20 m.

4.3.5 Simulation results

In what follows, we evaluate the performance of our approach in various scenarios with different characteristics.

Training analysis

We first consider a scenario with 2 UAVs and 4 targets in a 20×20 map. In particular, we perform multiple training phases of different duration; the longer training includes 3000 episodes, for a total of 300,000 training samples, which ensures that all our algorithms achieve convergence. The look-ahead approach is abbreviated as LA(4), as we set $n_\ell = 4$. This already had a significant

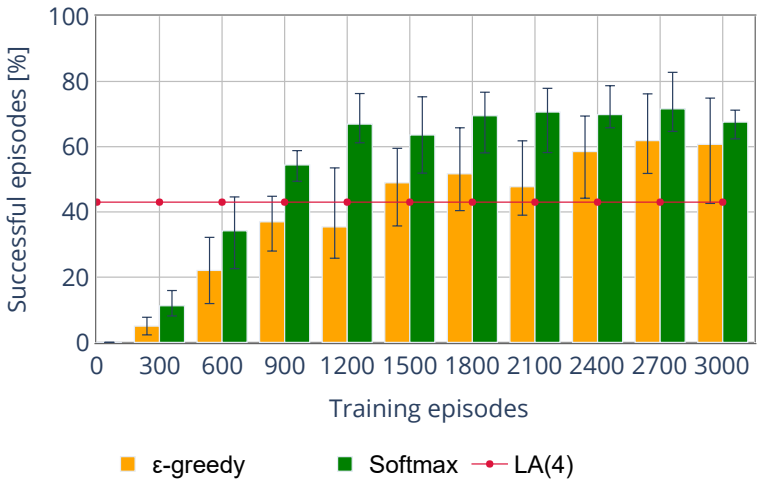


Figure 4.18: Success probability over the training phase in the *cluster* scenario with 2 UAVs.

computational cost, and in our simulation each look-ahead decision takes approximately 15 times longer than running a trained DDQL agent. We do not consider $n_\ell > 4$, since the computational cost of such a technique becomes excessive with limited performance gains: without coordination among the UAVs, which requires a prediction of the movements of other drones in the swarm, there is a limit on the performance of the swarm even with an infinite horizon. In some brief tests (which had to be on maps of a limited size due to the computational complexity of LA with a longer horizon), we noticed that LA(8) and even LA(12) show limited gains over LA(4), as the biggest factor in determining the speed at which the UAVs find the target becomes the coordination of the swarm once the horizon reaches 3 or 4 steps.

Fig. 4.18 shows the success probability in the *cluster* scenario as a function of the training set size and of the considered exploration profile and approach. DDQL combined with the softmax approach catches up with LA(4) in less than 900 training episodes, converging to a success probability between 0.65 and 0.7. The ε-greedy approach has a lower final performance and requires more time to converge with respect to the softmax profile. The error bars show the best and worst results over 5 test phases, showing that the performance improves as the UAVs gain more experience. The performance boost over the look-ahead approach is due to the DDQL scheme's ability to exploit the correlation among the target positions, quickly finding the other targets after the first one has been spotted. Instead, in the *sparse* scenario, the final performance of DDQL is similar to that of LA(4), as Fig. 4.19 shows. In general, both DDQL and LA(4) have more success than in the cluster scenario, as finding the scattered targets is easier than finding clusters in the limited duration of an episode.

Success rate over time

The next set of results refer to the performance of the strategy learned by the proposed framework. Fig. 4.20 reports the probability of one or both drones reaching the target as a function of the

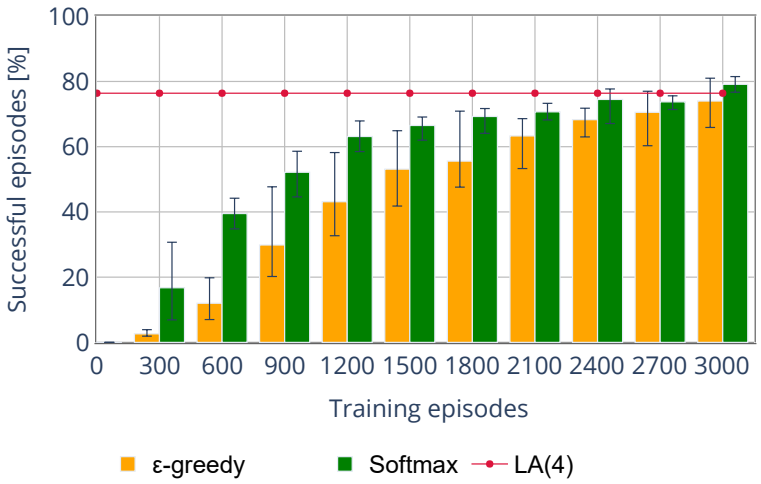
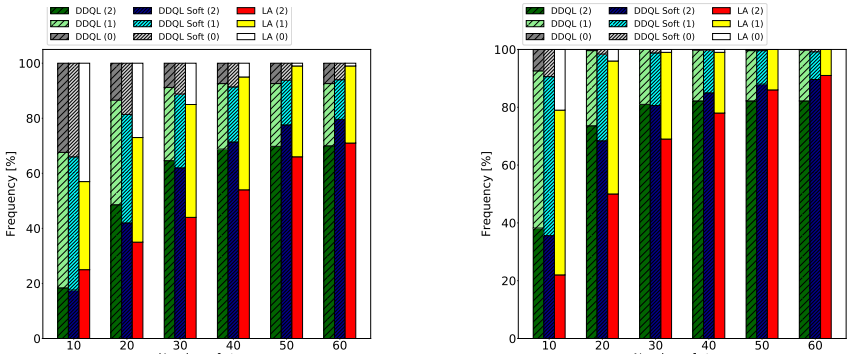


Figure 4.19: Success probability over the training phase in the *sparse* scenario with 2 UAVs.

number of steps. Therefore, the figure shows the trade-off between the time needed by UAVs to accomplishing their task and the success rate. In the cluster scenario (Fig. 4.20a), DDQL is much faster than LA, but its performance peaks out, and after 40 steps the probability of the UAVs reaching their targets does not change significantly. Indeed, we observed that, in certain cases, when a drone reaches the target, but the other one is far from any feature of the map, the latter can end up staying in place, as its Q-values for that scenario are not precise and all actions have a similar (low) value. This almost never happens before the first UAV reaches its target, since the change in the system state due to the movement of one UAV is generally enough to make the other UAV move. This is not a problem for LA, whose success rate keeps increasing with time; in the sparse scenario (Fig. 4.20b), LA even ends up reaching more targets than DDQL after 50 steps. The solution we found to avoid this roadblock is simply to maintain a low softmax temperature $\tau = 0.1$ even during the test phase: the bar chart shows that the DDQL Soft system is slightly slower than the greedy DDQL at the beginning, but it can avoid getting stuck. This randomization allows the agent to get out of loops, as sometimes a random sub-optimal action will change the state and allow it to reconsider, while the greedy system will keep performing the same action and remain in the same state. LA essentially does the same, randomizing its action when it is unsure which one is the best.

Fig. 4.21 shows one such situation: as one UAV has reached its target, while the other is far from any identified target, its Q-values will be very similar to each other, and some of the time it will stay motionless or move in small loops, as its state never changes. The fact that most of the map is still unexplored increases the probability of the UAV getting stuck, as it will have limited information and its Q-values will be very similar. In the following, all the results are referred to the DDQL Soft system with $\tau = 0.1$ unless otherwise stated.



(a) Cluster scenario (b) Sparse scenario

Figure 4.20: The bars indicate the probability mass distribution of the number of UAVs that successfully accomplish their task (i.e., hover upon a target) by the end of the episode, when varying the duration of the episode. Each group of bars refers to the performance achieved by DDQL (with and without softmax) and by LA, in the Cluster (a) and Sparse (b) scenarios, with a total of 4 targets and 2 UAVs.

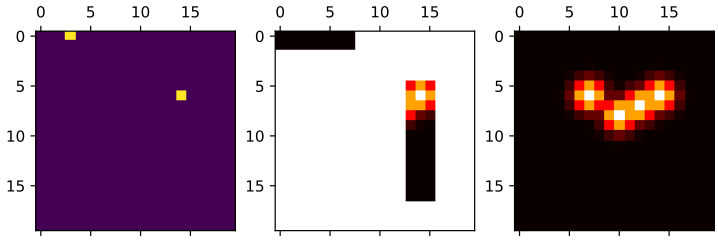
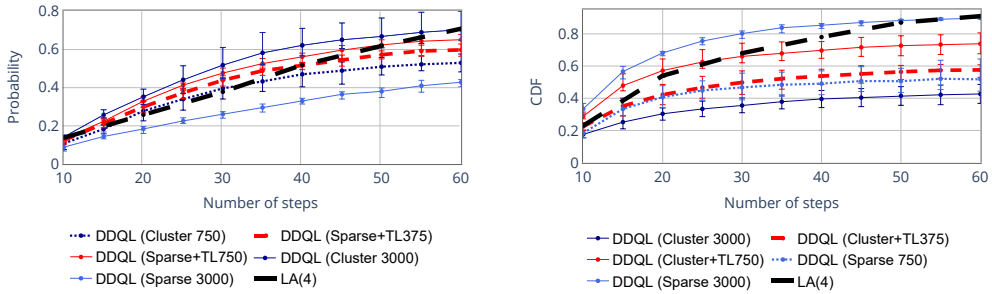


Figure 4.21: Example of an episode where the second UAV is not able to reach the cluster



(a) (b)

Figure 4.22: CDF of the episode duration for different algorithms in the cluster (a) and sparse (b) scenario with 2 UAVs.

Adaptability and transfer learning

Here we investigate the adaptability of the proposed DDQL scheme, and the potential of the transfer learning paradigm. The latter involves the execution of an additional training phase in a different scenario than the one seen during the initial training. To this end, we consider a common target scenario, i.e., cluster (or sparse), and compare the results achieved when using strategies learned in the other domain, i.e., sparse (or cluster). More specifically, we consider the following cases:

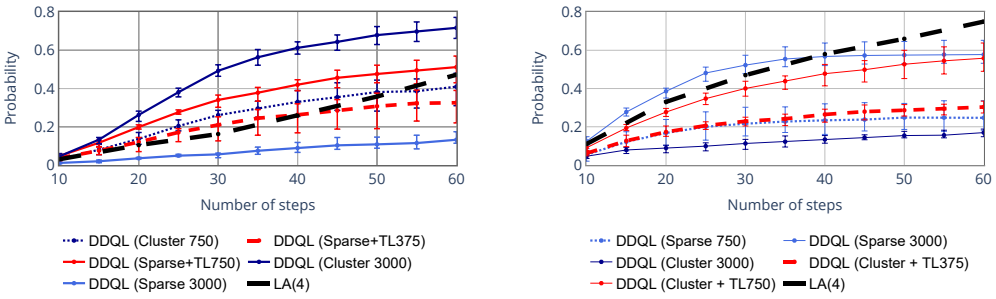
- "Cluster N_e": training on N_e episodes in the cluster scenario;
- "Sparse N_e ": training on N_e episodes in the sparse scenario;
- "Cluster+TL N_t": pre-training on $N_e = 3000$ episodes in the cluster scenario, followed by an additional training of N_t episodes in the target scenario.
- "Sparse+TL N_t": pre-training on $N_e = 3000$ episodes in the sparse scenario, followed by an additional training of N_t episodes in the target scenario.

Fig. 4.22a shows the cumulative distribution function (CDF) of the *episode duration*, defined as the time until all the drones reach targets or the testing episode limit (here fixed to 60 steps) is reached. We also report the results for LA with four steps, LA(4), as a benchmark. Each point is hence the probability that all drones have accomplished their task by a given number of steps.

We observe that, as expected, the Cluster strategy achieves the highest success probability with a limited number of steps. LA(4) can equal its performance only when the episode duration reaches the limit of 60 steps (i.e., in less than 30% of the cases). Instead, 750 episodes of training in the cluster scenario are not sufficient to outperform LA(4), but actually enough to outperform a model trained in the sparse scenario. However, a short retraining of such model in the correct (cluster) scenario allows the algorithm to get a significant performance boost, outperforming LA(4) and getting very close to the performance of the Cluster 3000 model, which is fully trained in the correct scenario and with more than twice the number of episodes.

We repeated the experiment by swapping the role of the sparse and cluster scenarios, and changing the number of episodes during the training phase, as reflected in the legend of Fig. 4.22b, which reports the results. As in the previous case, LA(4) meets the performance of DDQL only for episodes of 60 steps, i.e., in less than 15% of the cases. Transfer learning is again very effective, as a 750 episode re-training significantly boosts the baseline performance compared to starting from scratch. We highlight that, in general, the number of steps necessary to reach the targets is comparatively lower than in the previous scenario since, as already discussed, it is easier for UAVs to find targets in the sparse scenario.

Fig. 4.23a and Fig. 4.23b show the results for a scenario with 3 UAVs: in both cases, transfer learning is effective, but the performance is lower in the sparse scenario than in the cluster one. In this case, the risk of getting stuck is increased and the algorithm needs more training to perform effectively in all maps.



(a) (b)

Figure 4.23: CDF of the episode duration for different algorithms in the cluster (a) and sparse (b) scenario with 3 UAVs.

Obstacles

In what follows, we consider a modified version of the cluster scenario, where some obstacles are added to the map. In particular, we empirically set the percentage of the map occupied by obstacles to 10%, searching for a balance between increased system complexity and the realism of the scenario. An example of the system state representation with obstacles is shown in Fig. 4.24 at the beginning and at the end of an episode. The obstacles are marked in green.

Fig. 4.25a shows the performance of the LA approach and DDQL in the case of 2 UAVs and 4 targets. The DDQL solution has been trained for scenarios with 2, 3 and 4 UAVs (labeled in the plots as 2D, 3D, and 4D, respectively), and then tested in the scenario with 2 and 3 UAVs, with and without the use of the softmax approach in the testing phase. In both cases, it is clear that the models trained with more UAVs are able to outperform those with fewer UAVs in both considered scenarios. Furthermore, as for the case without obstacles, the use of the softmax policy during the testing phase increases the performance, especially when the episodes are longer, as it keeps the UAVs from getting stuck. In the scenario with 3 UAVs in Fig. 4.25b, the performance is generally lower, meaning that the swarm needs more training. However, DDQL is able to outperform the LA approach in both cases, reaching targets significantly faster in the scenario with 3 drones.

Transfer learning on bigger maps with larger swarms and communication impairments

We then show how well DDQL is able to generalize to bigger maps in the testing phase. For this reason, the algorithm has been trained on a map with $M = 24$, maintaining $F = 20$, and the testing phase included bigger maps and different numbers of clusters. All the results shown in the following figures are obtained with 100-step episodes: the longer duration is needed to allow the agents to reach the targets even in bigger maps. For similar reasons, the scenarios with more clusters are studied to maintain a similar proportion of surface occupied by targets even in the bigger maps. Fig. 4.26a and Fig. 4.26b show how the performance varies as a function of the

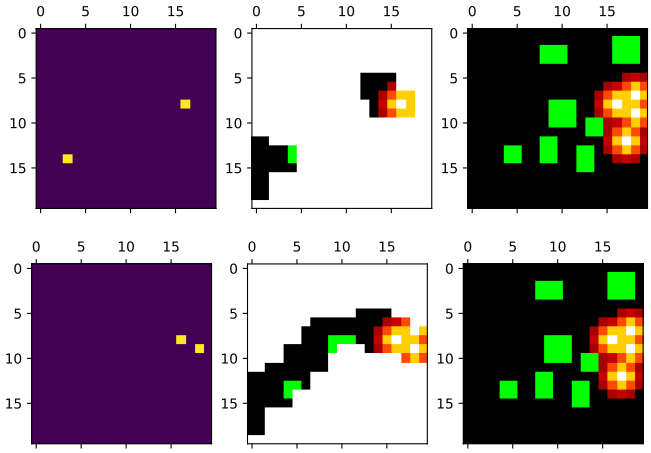
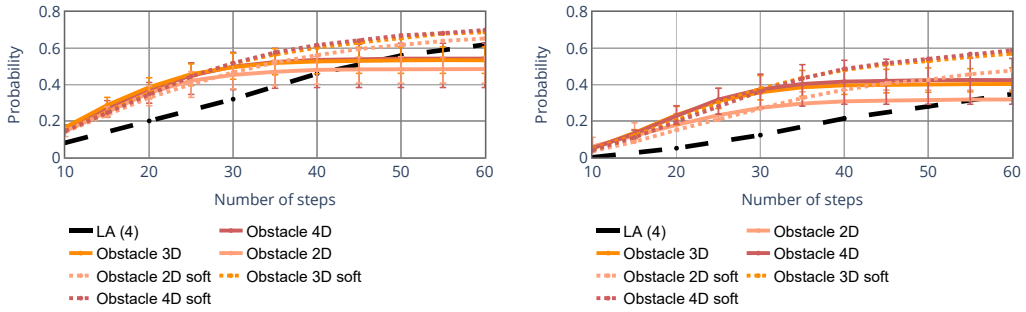
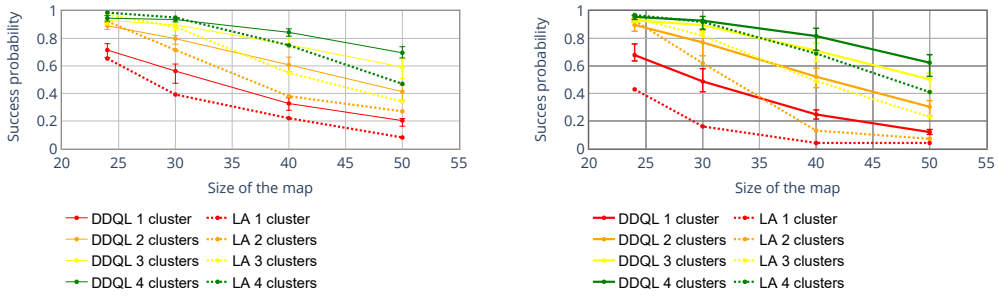


Figure 4.24: Drone positions (left), known map (center), real map (right). Beginning (above) and end (below) of an episode with obstacles.



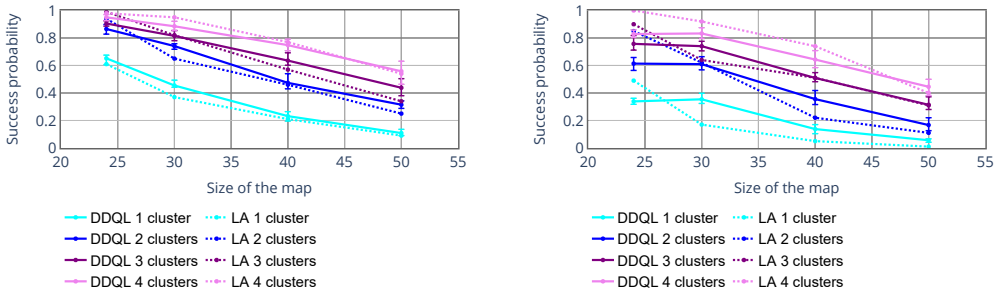
(a) 2 UAVs. (b) 3 UAVs.

Figure 4.25: CDF of the episode duration for different algorithms in the obstacle scenario.



(a) 2 UAVs. (b) 3 UAVs.

Figure 4.26: Success probability as a function of the map size and the number of clusters.



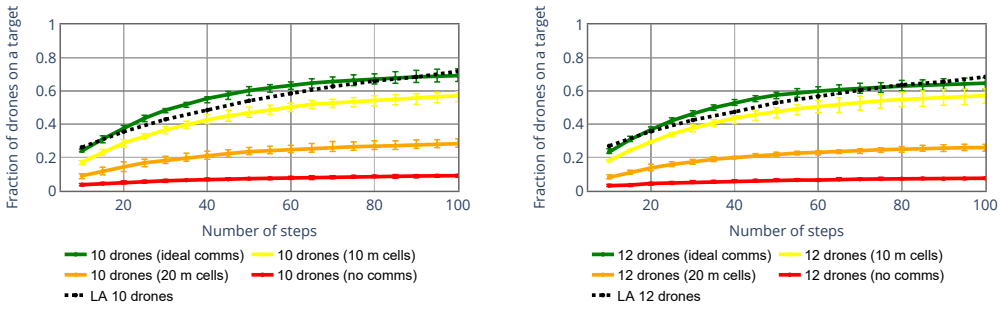
(a) 2 UAVs.

(b) 3 UAVs.

Figure 4.27: Success probability as a function of the map size and the number of clusters with obstacles.

size of the environment and the number of clusters present in the map. In both cases, DDQL shows a good adaptability, getting better performance than LA in all cases, with a bigger gain in bigger maps. In Fig. 4.27a and Fig. 4.27b, the same scenarios are studied with the addition of the obstacles in the map, covering about 10% of the size of the map. In this case, DDQL will need some retraining to reach LA's performance on smaller maps, while the performance is similar when the map is bigger. However, we recall that DDQL also has a significant advantage in terms of computational cost, so it is preferable if performance is similar.

It is also interesting to test the transfer capabilities of the algorithms in more complex scenarios, including far larger swarms and imperfect communications: as DDQL relies on information from other UAVs to find targets and avoid collisions, a limited communication range can impair its performance significantly. As Fig. 4.28a shows, 10 drones moving in a large map with obstacles (with 16 targets in 4 clusters, as above) can coordinate effectively with no retraining, outperforming the LA approach. Performance loss is limited even with communication restrictions if each cell is a square with a 10 m side, corresponding to a maximum range of about 11 cells with 50% packet loss at the boundary of the coverage area. Performance loss with respect to the perfect communication scenario is limited, confirming the intuitive idea that information from neighbors inside the visible area is the most critical to find and reach the targets. If the cell side is doubled, effectively halving the communication range and introducing significant errors even for packets between immediate neighbors, the performance drops significantly, and becomes even worse if there is no communication at all between the UAVs. This would be true for any cooperative algorithm, as information from other agents can be used to optimize the exploration of the map, but we highlight that DDQL has always been trained assuming ideal communication, and the communication impairments have been considered only in the test phase. Therefore, the UAVs might be confused by the lack of information, and a partial retraining might yield better results as the agents transfer their experience and learn to deal with the more limited feedback. On the other hand, the algorithm scales extremely well to larger swarms, slightly outperforming LA even with no retraining in the new scenario. The same pattern holds for the case with 12 drones, which is shown in Fig. 4.28b.



(a) Performance of a swarm of 10 UAVs.

(b) Performance of a swarm of 12 UAVs.

Figure 4.28: Effect of imperfect communications on the performance of DDQL in a large map.

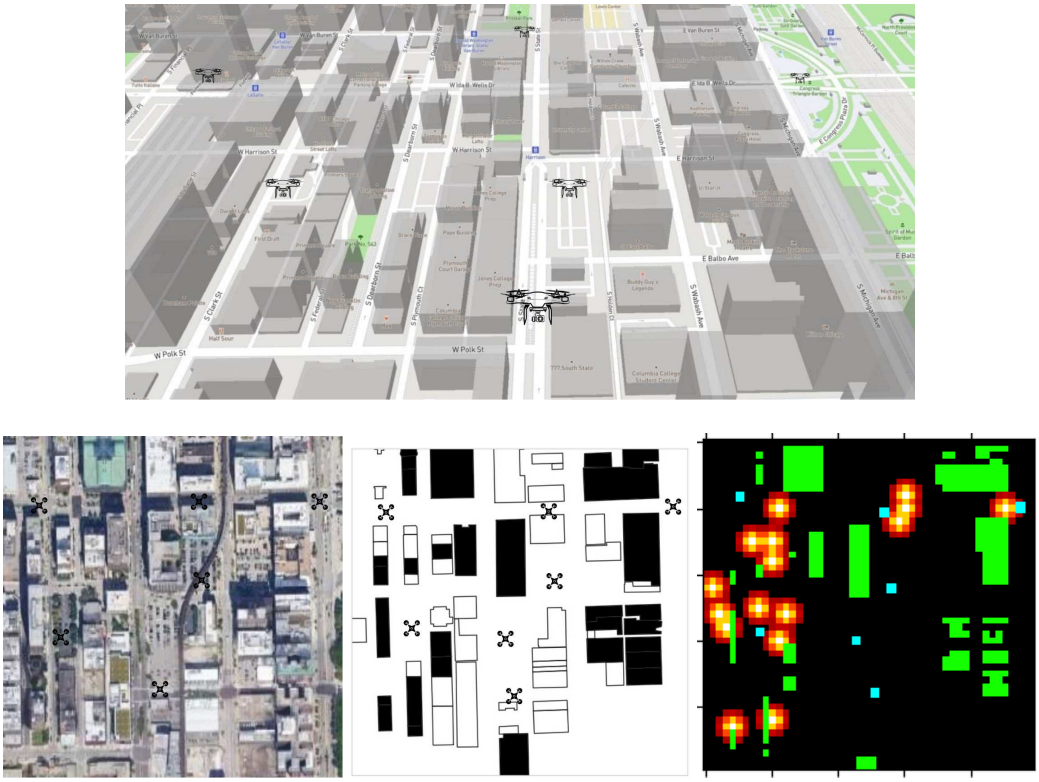


Figure 4.29: Extraction of the map from building height data in a 500 m by 500 m area in the downtown Chicago Loop neighborhood.

Finally, we tested an extreme transfer learning scenario, not only increasing the size of the map and the number of UAVs, but also switching from the synthetic obstacle distribution on the map to one derived from a real map. The map of obstacles was obtained from a city map of the area just east of LaSalle Street Station in downtown Chicago, in the central Loop neighborhood. As shown in Fig. 4.29, we obtained the height profiles of the buildings in the area, considering as

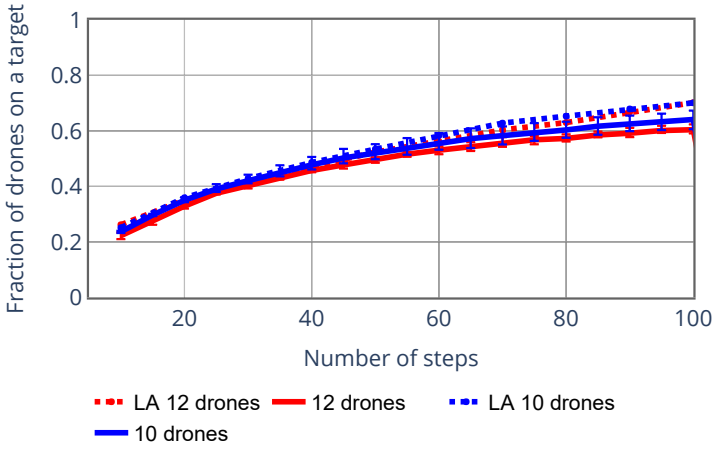


Figure 4.30: Performances on the real map of Chicago

obstacles their parts with a height of over 10 stories (i.e., approximately 40 m, the minimum legal hovering altitude for UAVs). The 500 m by 500 m area was then divided into 2500 square cells with 10 m sides, converting the height profile to an obstacle map in the grid. 11% of the map was occupied by obstacles, so the map was approximately as full as the one used in the training, which had 10% obstacle cover, but the individual obstacles were larger and concentrated along South Wabash Avenue and South Dearborn Street. This is an additional hurdle for DDQL, which was not trained to deal with obstacles concentrated along streets, which make it more difficult to find an appropriate path. However, as Fig. 4.30 shows, the DDQL system can find targets approximately as fast as LA, underperforming a little only on higher percentiles. With a modicum of retraining, DDQL should be able to adapt to the different structure, exploiting the regularities in city blocks to avoid obstacles and find targets even more quickly.

In conclusion, we have shown that DDQL is able to find efficient strategies for the UAVs to reach targets faster than look-ahead solutions in complex environments. The algorithm is scalable to larger maps, larger swarms, and limited communications without any retraining, and can deal with obstacles and very different target distributions with a limited amount of retraining. This shows that the solution is powerful and versatile, adapting easily to new conditions. However, there is still some margin for improvement, particularly in scenarios in which almost all the UAVs in the swarm have already reached a target, while the last stragglers are far from any feature in the map. This case represents most of the residual failures of the algorithm, and solving it is an important future objective.

4.3.6 Conclusions and future work

In this work, we studied the problem of area monitoring and surveillance with a swarm of drones. We modeled the environment with a 2D grid and cast the problem into the theoretical framework of ND-POMDP. We have examined various scenarios, including obstacles and maps of different

sizes, and the proposed algorithm outperformed a computationally intensive look-ahead approach in almost all scenarios.

Important research directions include the introduction of dynamic targets, which would be an important step to increase the scenario's realism, as well as different roles for the drones, which can be assigned dynamically and would allow us to examine another interesting aspect of the MARL problem, increasing the difficulty of coordinating the UAVs' actions.

5 Conclusion

With the always increasing number of machine learning applications and the exponential growth of the size of modern neural networks, orchestrating distributed training and inference is becoming very demanding from an infrastructure standpoint. Specifically, the substrate of distributed learning is represented by the underlying communication network, which moves data and models from one point to the other. It is indeed envisioned that, in the near future, learning-oriented data will flood the networks, producing a substantial portion of the data traffic. Consequently, it is of paramount importance that the networks of tomorrow, which will be an integral part of the learning systems, will be designed with these objectives in mind, and that learning and communication protocols will be jointly optimized to better cooperate and utilize the physical resources.

To this end, this thesis is one of the very first attempts that tries to investigate the role that the information has in different distributed learning scenarios, exploiting the inherent redundancy to minimize the need for network resources, without substantially damaging the performance.

Specifically, in this thesis we first focused on the setting of federated learning (FL), which is becoming the standard to perform privacy-preserving distributed learning, where by design the only information exchanged in the network is represented by model and gradient updates, and not by raw data. The impact of the wireless channel condition on the training convergence was empirically evaluated, and a way to trade off the amount of information for convergence speed is proposed [186]. We then described `FedPM` [200], which is a novel proposal that reduces the communication requirements by training only the neural networks' topology, without changing the specific values of the coefficients. Following on this, with `KLMS` [197] we designed a new way to compress and code model updates in FL, which is inspired by relative entropy coding and the importance sampling algorithm, and that dramatically outperforms state-of-the-art baselines in the rate-accuracy curve. In the end, we tried to generalize the idea and provided a theoretical formulation for the problem of communicating learning models, like neural networks, and analyzed some fundamental limits through the lens of information theory [198].

The focus was then moved to the analysis of distributed decision processes, where standard formulation of multi-armed bandit (MAB) and reinforcement learning (RL) have been extended to account for multiple agents and noisy/constrained communication channels. In particular, we physically separated the state observation process from that of taking actions, introducing a communication channel in between, which demands to carefully analyze how much of the state information is required to optimally interact with the environment. For the more tractable setting of contextual multi-armed bandit (CMAB) we provided the theoretical limits of the problem, identifying the minimum rate needed to solve the learning task [189, 187]. Also, we designed both theoretical and practical policy compression schemes which optimally trade off rate for performance. The same extension is then investigated in the more complex scenario of Deep Reinforcement Learning (DRL), for which a practical but very general and powerful coding scheme based on Vector Quantized Variational Autoencoder (VQ-VAE) was proposed and analyzed [234].

The last part of the thesis was then application-oriented, and some examples on how distributed learning algorithms can be used in real-world cases were reported. In the first two use cases, which were part of a joint collaboration with the University of Bologna and Huawei Research Center in Munich, particular variants of the multi-agent CMAB were proposed and solved to optimize distributed resource allocation schemes in Industrial Internet of Things (IIoT), with the aim of providing Ultra-Reliable and Low-Latency Communications (URLLC) [194]. Thanks to the proposed learning solutions, the User Equipments (UEs) in the system were able to cooperate while sharing the same wireless resources for uplink communications without incurring collisions. Then, a multi-agent reinforcement learning (MARL) framework was developed to design a control policy for a swarm of drones that need to collaboratively monitor an area with targets of interest [249]. Again, drones equipped with a DRL-based policy outperformed those with standard heuristic approaches in the evaluated metrics.

5.1 Final Considerations & Future Directions

On a broader perspective, the goal of this thesis was to push to the limit the performance of distributed learning while minimizing the utilization of network resources. The main findings, both theoretical and empirical, suggest that the state-of-the-art algorithms that are usually adopted today do not optimally exploit the fundamental trade-off between communication rate and achieved performance, leaving room for improvements and research.

For example, when designing `FedPM` (see Section 2.3), the idea was not to consider vanilla FL and then to apply some compression methods on top of it, but rather to come up with a radically new training pipeline, i.e., optimize only the topology of a random network, which is efficient by first principles. This way, even though `FedPM` will not achieve the best overall performance when considering limitless bandwidth, it does in the ~ 1 bit per parameter (bpp) regime, as no compression has to be performed on its updates, which contain by design less information than standard real-valued coefficients. This is also true for `KLMS` (see Section 2.4), which combines recent information-theoretic results and the side information known (for free) by the server to further increase by one order of magnitude the gain already provided by `FedPM`. Also, the analysis

performed in Section 3.2 shows that the information redundancy during the training stage of MAB agents is not always necessary to optimize for the best policy. All these improvements were made possible because the distributed training schemes account for the communication perspective in the first principles.

To conclude, there is still a lot of work to do in the field of distributed and efficient machine learning, where not only communication but also privacy, security, computational and storage constraints will come into play. Even though it is hard to envision a fully distributed and edge machine learning world, the winning scheme will probably be a hybrid approach, as recently reported by Qualcomm [1], a pioneer in this field. Indeed, if huge and complex foundation models will still run in the cloud, edge AI and distributed training must be there to sustain on-device applications and exploit the amount of data collectively acquired by the end users. To this end, the research community should provide the necessary tools to the industry to understand *when* a specific task can be fully accomplished at the edge, or needs help from a more powerful cloud infrastructure in terms of data and/or computational capacity; then, once this is solved, it has to provide the necessary algorithms that can squeeze data and models to obtain the maximum performance with the minimum requirements, distilling cloud knowledge to the edge devices, when the specific task admits such approximation.

[1] https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/Whitepaper-The-future-of-AI-is-hybrid-Part-1-Unlocking-the-generative-AI-future-with-on-device-and-hybrid-AI.pdf

References

[1] 3GPP. “Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures”. In: *TS 36.213* (2021).

[2] 3GPP. “NR - Physical channels and modulation (Release 15)”. In: *TS 38.211* (2018).

[3] 3GPP. “NR - Physical layer procedures for data (Release 15)”. In: *TS 38.214* (2018).

[4] 3GPP. “NR and NG-RAN Overall Description (Release 15)”. In: *TS 38.300* (2018).

[5] 3GPP. *NR; Medium Access Control (MAC) protocol specification – Release 15*. Technical Specification (TS) 38.321. 3GPP, 2019.

[6] 3GPP. “Semi-Persistent Scheduling for 5G New Radio URLLC”. In: *R1-167309* (2016).

[7] 3GPP. “Service requirements for cyber-physical control applications in vertical domains; Stage 1 (Release 18)”. In: *TS 22.104* (2021).

[8] 3GPP. “Study on channel model for frequencies from 0.5 to 100 GHz (Release 16)”. In: *TR 38.901* (2019).

[9] 3GPP. “Study on NR Industrial Internet of Things (IoT) (Release 16)”. In: *TS 38.825* (2019).

[10] 5G-ACIA. “5G for Automation in Industry: Primary use cases, functions and service requirements”. In: *White Paper* (2019).

[11] 5G-ACIA. “5G for Connected Industries and Automation”. In: *Verband der Elektro- und Digitalindustrie (ZVEI) White Paper* (Feb. 2019).

[12] 5G-ACIA. “5G for Industrial Internet of Things (IIoT): Capabilities, Features, and Potential”. In: *ZVEI* (Nov. 2021).

[13] 5G-ACIA. “Integration of Industrial Ethernet Networks with 5G networks”. In: *Verband der Elektro- und Digitalindustrie (ZVEI) White Paper* (Nov. 2019).

[14] 5G-Clarity. “Use Case Specifications and Requirements”. In: *Verband der Elektro- und Digitalindustrie (ZVEI) White Paper* (Mar. 2020).

[15] Martin Abadi et al. “Deep learning with differential privacy”. In: *Proceedings of the ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[16] Alessandro Achille et al. “The information complexity of learning tasks, their structure and their distance”. In: *Information and Inference: A Journal of the IMA* 10.1 (Jan. 2021), pp. 51–72.

[17] Mridul Agarwal, Vaneet Aggarwal, and Kamyar Azizzadenesheli. “Multi-Agent Multi-Armed Bandits with Limited Communication”. In: *arXiv:2102.08462 [cs]*. 2021.

[18] Naman Agarwal, Peter Kairouz, and Ziyu Liu. “The skellam mechanism for differentially private federated learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5052–5064.

[19] Shipra Agrawal and Navin Goyal. “Thompson sampling for contextual bandits with linear payoffs”. In: *International Conference on Machine Learning (ICML)* (2013).

[20] Alham Aji and Kenneth Heafield. "Sparse Communication for Distributed Gradient Descent". In: *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL). 2017.

[21] Maxwell M Aladago and Lorenzo Torresani. "Slot machines: Discovering winning combinations of random weights in neural networks". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 163–174.

[22] Dario Albani, Daniele Nardi, and Vito Trianni. "Field coverage and weed mapping by UAV swarms". In: *International Conference on Intelligent Robots and Systems*. IEEE. Sept. 2017, pp. 4319–4325.

[23] Hande Alemdar et al. "Ternary neural networks for resource-efficient AI applications". In: *International Joint conference on Neural Networks (IJCNN)*. 2017, pp. 2547–2554.

[24] Dan Alistarh et al. "QSGD: Communication-efficient SGD via gradient quantization and encoding". In: *Advances in Neural Information Processing Systems* (2017).

[25] Shun-ichi Amari. "Backpropagation and Stochastic Gradient Descent Method". In: *Neurocomputing* (1993), pp. 185–196.

[26] M. M. Amiri et al. "Convergence of Update Aware Device Scheduling for Federated Learning at the Wireless Edge". In: *IEEE Transactions on Wireless Communications [Early Access]* (2021).

[27] Mohammad Mohammadi Amiri et al. "Federated Learning With Quantized Global Model Updates". In: *arXiv preprint arXiv:2006.10672v2* (2020).

[28] Galen Andrew et al. "Differentially private learning with adaptive clipping". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17455–17466.

[29] Marko Angjelichinoski, Floe Trillingsgaard, and Petar Popovski. "A statistical learning approach to ultra-reliable low latency communication". In: *IEEE Transactions on Communications* 67.7 (Mar. 2019), pp. 5153–5166. DOI: 10.1109/TCOMM.2019.2907241.

[30] Samin Yeasar Arnob et al. "Single-Shot Pruning for Offline Reinforcement Learning". In: *Neural Information Processing Systems Workshop in Offline Reinforcement Learning* (2021).

[31] Mohammad Babaeizadeh et al. "GA3C: GPU-based A3C for Deep Reinforcement Learning". In: *Neural Information Processing Systems Workshop* (2016).

[32] Sara Babakniya et al. "Federated Sparse Training: Lottery Aware Model Compression for Resource Constrained Edge". In: *arXiv preprint arXiv:2208.13092* (2022).

[33] T. Bai and R. W. Heath. "Coverage and Rate Analysis for Millimeter-Wave Cellular Networks". In: *IEEE Transactions on Wireless Communications* 14.2 (Oct. 2015), pp. 1100–1114. DOI: 10.1109/TWC.2014.2364267.

[34] David Baldazo, Juan Parras, and Santiago Zazo. "Decentralized multi-agent deep reinforcement learning in swarms of drones for flood monitoring". In: *European Signal Processing Conference (EUSIPCO)*. EURASIP. Sept. 2019. ISBN: 9789082797039.

[35] B. Balle, G Barthe, and M. Gaboardi. "Privacy amplification by subsampling: tight analyses via couplings and divergences". In: *Advances in neural information processing systems* (2018).

[36] Borja Balle et al. "Privacy Amplification via Random Check-Ins". In: *Advances in Neural Information Processing Systems*. 2020.

[37] Leighton Pate Barnes et al. "rTop-k: A statistical estimation approach to distributed SGD". In: *IEEE Journal on Selected Areas in Information Theory* 1.3 (Nov. 2020), pp. 897–907.

[38] Andrew Barron, Jorma Rissanen, and Bin Yu. "The minimum description length principle in coding and modeling". In: *IEEE transactions on information theory* 44.6 (1998), pp. 2743–2760.

[39] Ran Ben Basat et al. "QUICK-FL: Quick Unbiased Compression for Federated Learning". In: *arXiv preprint arXiv:2205.13341* (2022).

[40] Raef Bassily et al. "Learners that Use Little Information". In: *Proceedings of Algorithmic Learning Theory*. Vol. 83. Apr. 2018, pp. 25–55.

[41] Edgar Beck, Carsten Bockelmann, and Armin Dekorsy. "Semantic Communication: An Information Bottleneck View". In: *arXiv:2204.13366* (Apr. 2022).

[42] Richard Bellman. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684.

[43] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[44] Jeremy Bernstein et al. "signSGD: Compressed optimisation for non-convex problems". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 560–569.

[45] Simone Bianco et al. "Benchmark analysis of representative deep neural network architectures". In: *IEEE Access* 6 (2018), pp. 64270–64277.

[46] Sameer Bibikar et al. "Federated dynamic sparse training: Computing less, communicating less, yet learning better". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6080–6088.

[47] Davis Blalock et al. "What Is The State Of Neural Network Pruning?" In: *Proceedings of the 3^{rd} MLSys Conference* (2021).

[48] Mate Boban, Marco Giordani, and Michele Zorzi. "Predictive Quality of Service (PQoS): The Next Frontier for Fully Autonomous Systems". In: *IEEE Network* 35.6 (Nov. 2021), pp. 104–110.

[49] Rémi Bonnefoi et al. "Multi-Armed Bandit Learning in IoT Networks: Learning Helps Even in Non-stationary Settings". In: *Cognitive Radio Oriented Wireless Networks*. 2018, pp. 173–185.

[50] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. "Deep Joint Source-Channel Coding for Wireless Image Transmission". In: *IEEE T. Cognitive Commun. Networking* 5.3 (May 2019), pp. 567–579.

[51] Craig Boutilier et al. "Differentiable Meta-Learning of Bandit Policies". In: *Advances in Neural Information Processing Systems*. 2020.

[52] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "A Comprehensive Survey of Multiagent Reinforcement Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics* 38.2 (Mar. 2008), pp. 156–172. DOI: 10.1109/TSMCC.2007.913919.

[53] Lucian Busoniu, Robert Babuška, and Bart De Schutter. "Multi-agent Reinforcement Learning: An Overview". In: *Innovations in Multi-Agent Systems and Applications* 310 (Nov. 2010), pp. 113–147.

[54] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. "Multi-agent Reinforcement Learning: An Overview". In: *Innovations in Multi-Agent Systems and Applications* (2010), pp. 183–221.

[55] Baturalp Buyukates and Sennur Ulukus. “Timely Communication in Federated Learning”. In: *arXiv preprint arXiv:2012.15831* (2020).

[56] S. Cavallero et al. “A New Scheduler for URLLC in 5G NR IIoT Networks with Spatio-Temporal Traffic Correlations”. In: *IEEE International Conference on Communications (ICC)*. 2023.

[57] Ursula Challita, Walid Saad, and Christian Bettstetter. “Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs”. In: *International Conference on Communications (ICC)*. IEEE. May 2018.

[58] Sourav Chatterjee and Persi Diaconis. “The sample size required in importance sampling”. In: *The Annals of Applied Probability* 28.2 (2018), pp. 1099–1135.

[59] Hong-You Chen and Wei-Lun Chao. “Fed{BE}: Making Bayesian Model Ensemble Applicable to Federated Learning”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=dgtpE6gKjHn.

[60] Jiasi Chen and Xukan Ran. “Deep Learning With Edge Computing: A Review”. In: *Proceedings of the IEEE* 107.8 (July 2019), pp. 1655–1674. DOI: 10.1109/JPROC.2019.2921977.

[61] Lixing Chen, Jie Xu, and Zhuo Lu. “Contextual Combinatorial Multi-armed Bandits with Volatile Arms and Submodular Reward”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[62] M. Chen et al. “Convergence Time Optimization of Federated Learning over Wireless Networks”. In: *IEEE Transactions on Wireless Communications [Early Access]* (2020). DOI: 10.1109/TWC.2020.3042530.

[63] Chien-Yao Wang and Alexey Bochkovskiy and Hong-Yuan Mark Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)* (2023).

[64] Soon Jo Chung et al. “A Survey on Aerial Swarm Robotics”. In: *IEEE Transactions on Robotics* 34.4 (Aug. 2018), pp. 837–855. ISSN: 15523098.

[65] A. V. Clemente, H. N. Castejón, and A. Chandra. “Efficient Parallel Methods for Deep Reinforcement Learning”. In: *ArXiv e-prints* (May 2017). arXiv: 1705.04862 [cs.LG].

[66] Gregory Cohen et al. “EMNIST: Extending MNIST to handwritten letters”. In: *International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 2921–2926.

[67] Anne G. E. Collins and Michael J. Frank. “How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis”. In: *European Journal of Neuroscience* 35.7 (Apr. 2012), pp. 1024–1035.

[68] Jade Copet et al. “Simple and Controllable Music Generation”. In: *arXiv preprint arXiv:2306.05284* (2023).

[69] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[70] Imre Csiszár and Gabor Tusnády. “Information Geometry and Alternating Minimization Procedures”. In: *Statistics and Decisions, Supplement Issue* (Jan. 1984), pp. 205–237.

[71] Paul Warner Cuff, Haim H. Permuter, and Thomas M. Cover. “Coordination Capacity”. In: *IEEE Transactions on Information Theory* 56.9 (Sept. 2010), pp. 4181–4206. DOI: 10.1109/TIT.2010.2054651.

[72] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. "Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks". In: *IEEE Transactions on Wireless Communications* 19.2 (Feb. 2020), pp. 729–743. ISSN: 15582248.

[73] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. "The application of multi-agent reinforcement learning in UAV networks". In: *International Conference on Communications Workshops (ICC)*. IEEE. May 2019. ISBN: 9781728123738.

[74] G. Cuozzo et al. "Enabling URLLC in 5G NR IIoT Networks: A Full-Stack End-to-End Analysis". In: *Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 2022.

[75] Rong Dai et al. "DisPFL: Towards Communication-Efficient Personalized Federated Learning via Decentralized Sparse Training". In: *arXiv preprint arXiv:2206.00187* (2022).

[76] Ofir David, Shay Moran, and Amir Yehudayoff. "On Statistical Learning via the Lens of Compression". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2792–2800. ISBN: 9781510838819.

[77] Li Deng. "The MNIST database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[78] James Diffenderfer and Bhavya Kailkhura. "Multi-Prize Lottery Ticket Hypothesis: Finding Accurate Binary Neural Networks by Pruning A Randomly Weighted Network". In: *International Conference on Learning Representations*. 2020.

[79] G. D. Durgin, T. S. Rappaport, and D. A. de Wolf. "New analytical models and probability density functions for fading in wireless communications". In: *IEEE Transactions on Communications* 50.6 (Aug. 2002), pp. 1005–1015. DOI: 10.1109/TCOMM.2002.1010620.

[80] Cynthia Dwork et al. "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[81] Khaoula El Mekkaoui et al. "Distributed stochastic gradient MCMC for federated learning". In: *arXiv preprint arXiv:2004.11231* (2020).

[82] Khaoula El Mekkaoui et al. "Federated Stochastic Gradient Langevin Dynamics". In: *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 1703–1712.

[83] Peter Elias. "Universal codeword sets and representations of the integers". In: *IEEE Transactions on Information Theory* 21.2 (Mar. 1975), pp. 194–203.

[84] Úlfar Erlingsson et al. "Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2019, pp. 2468–2479.

[85] ETSI. "Study on 5G NR User Equipment (UE) application layer data throughput performance". In: *TR 137 901-5* (2020).

[86] Marco Federici et al. "Learning Robust Representations via Multi-View Information Bottleneck". In: *International Conference on Learning Representations* abs/2002.07017 (2020).

[87] Vitaly Feldman, Audra McMillan, and Kunal Talwar. "Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 954–964. DOI: 10.1109/FOCS52979.2021.00096.

[88] Paulo Abelha Ferreira et al. “Bayesian SignSGD Optimizer for Federated Learning”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[89] Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. “Compressing images by encoding their latent representations with relative entropy coding”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16131–16141.

[90] Gergely Flamich, Stratis Markou, and José Miguel Hernández-Lobato. “Fast relative entropy coding with a* coding”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 6548–6577.

[91] Jakob N Foerster et al. “Learning to communicate with Deep multi-agent reinforcement learning”. In: *30th Int. Conf. Neural Inf. Proc. Sys. (NeurIPS)*. Dec. 2016.

[92] Jakob N. Foerster et al. “Learning to Communicate with Deep Multi-Agent Reinforcement Learning”. In: *arXiv:1605.06676 [cs]* (May 2016). arXiv: 1605.06676.

[93] Emmanouil Fountoulakis, Nikolaos Pappas, and Marios Kountouris. “Goal-oriented policies for cost of actuation error minimization in wireless autonomous systems”. In: *IEEE Commun. Lett. (Early Access)* (June 2023).

[94] Jonathan Frankle and Michael Carbin. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks”. In: *International Conference on Learning Representations*. 2018.

[95] B. J. Frey and G. E. Hinton. “Efficient Stochastic Source Coding and an Application to a Bayesian Network Source Model”. In: *The Computer Journal* 40 (Jan. 1997), pp. 157–165.

[96] Marco Giordani et al. “Toward 6G Networks: Use Cases and Technologies”. In: *IEEE Communications Magazine* 58.3 (Mar. 2020), pp. 55–61.

[97] Antonious M. Girgis, Deepesh Data, and Suhas Diggavi. “Differentially Private Federated Learning with Shuffling and Client Self-Sampling”. In: *2021 IEEE International Symposium on Information Theory (ISIT)*. 2021, pp. 338–343. DOI: 10.1109/ISIT45174.2021.9517906.

[98] Anirudh Goyal et al. “Transfer and Exploration via the Information Bottleneck”. In: *International Conference on Learning Representations*. 2019.

[99] Zhouyou Gu et al. “Knowledge-Assisted Deep Reinforcement Learning in 5G Scheduler Design: From Theoretical Framework to Implementation”. In: *IEEE Journal on Selected Areas in Communications* 39.7 (May 2021), pp. 2014–2028. DOI: 10.1109/JSAC.2021.3078498.

[100] Deniz Gündüz et al. “Beyond Transmitting Bits: Context, Semantics, and Task-Oriented Communications”. In: *IEEE Journal on Selected Areas in Communications* 41.1 (Nov. 2023), pp. 5–41. DOI: 10.1109/JSAC.2022.3223408.

[101] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *International Conference on Machine Learning (ICML)* (2018).

[102] Hassan Halabian. “Distributed resource allocation optimization in 5G virtualized networks”. In: *IEEE Journal on Selected Areas in Communications* 37.3 (Feb. 2019), pp. 627–642.

[103] Song Han, Huizi Mao, and William J. Dally. “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding”. In: *International Conference on Learning Representations ICLR* (2016).

[104] Osama A. Hanna, Lin F. Yang, and Christina Fragouli. "Solving Multi-Arm Bandit Using a Few Bits of Communication". In: *38 th International Conference on Machine Learning*. 2021.

[105] Prahladh Harsha et al. "The Communication Complexity of Correlation". In: *IEEE Transactions on Information Theory* 56.1 (Jan. 2010), pp. 438–449. DOI: 10.1109/TIT.2009.2034824.

[106] Prahladh Harsha et al. "The communication complexity of correlation". In: *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*. IEEE. 2007, pp. 10–23.

[107] Burak Hasircioglu and Deniz Gunduz. "Privacy Amplification via Random Participation in Federated Learning". In: (2022). DOI: 10.48550/ARXIV.2205.01556. URL: https://arxiv.org/abs/2205.01556.

[108] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. "Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=r1f0YiCctm.

[109] Serhii Havrylov and Ivan Titov. "Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols". en. In: *Neural Information Processing Systems (NIPS)* (2017).

[110] K. He and J. Sun. "Convolutional neural networks at constrained time cost". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 5353–5360. DOI: 10.1109/CVPR.2015.7299173.

[111] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[112] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[113] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. "A survey and critique of multiagent deep reinforcement learning". In: *Autonomous Agents and Multi-Agent Systems* 33.6 (Nov. 2019), pp. 750–797. ISSN: 15737454.

[114] Pablo Hernandez-Leal et al. "A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity". In: *3rd International Workshop on Conflict Resolution in Decision Making (COREDEMA)*. July 2017.

[115] Naser Hossein Motlagh, Tarik Taleb, and Osama Arouk. "Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives". In: *IEEE Internet of Things Journal* 3.6 (Sept. 2016), pp. 899–922. ISSN: 23274662.

[116] Seyyedali Hosseinalipour et al. "From federated learning to fog learning: Towards large-scale distributed machine learning in heterogeneous wireless networks". In: *arXiv preprint arXiv:2006.03594* (2020).

[117] Yu-Hsin Hsu and Rung-Hung Gau. "Reinforcement Learning-based Collision Avoidance and Optimal Trajectory Planning in UAV Communication Networks". In: *IEEE Transactions on Mobile Computing* (June 2020).

[118] Jingzhi Hu et al. "Reinforcement learning for a cellular internet of UAVs: protocol design, trajectory control, and resource management". In: *IEEE Wireless Communications* 27.1 (Feb. 2020), pp. 116–123.

[119] Ruiquan Huang et al. “Federated Linear Contextual Bandits”. In: *35th Conference on Neural Information Processing Systems (NeurIPS)* (2021).

[120] Fatima Hussain et al. “Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges”. In: *IEEE Communications Surveys & Tutorials* 22.2 (Jan. 2020), pp. 1251–1275.

[121] Maximilian Igl et al. “Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 13956–13968.

[122] Jacob Imola and Kamalika Chaudhuri. “Privacy amplification via bernoulli sampling”. In: *arXiv preprint arXiv:2105.10594* (2021).

[123] Berivan Isik, Tsachy Weissman, and Albert No. “An information-theoretic justification for model pruning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 3821–3846.

[124] Mikolaj Jankowski, Deniz Gündüz, and Krystian Mikolajczyk. “AirNet: Neural Network Transmission over the Air”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2022, pp. 2451–2456. DOI: 10.1109/ISIT50566.2022.9834372.

[125] Divyansh Jhunjhunwala et al. “Leveraging spatial and temporal correlations in sparsified mean estimation”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 14280–14292.

[126] Shaoxiong Ji et al. “Dynamic sampling and selective masking for communication-efficient federated learning”. In: *arXiv preprint arXiv:2003.09603* (2020).

[127] Yuang Jiang et al. “Model pruning enables efficient federated learning on edge devices”. In: *IEEE Transactions on Neural Networks and Learning Systems* (Apr. 2022).

[128] Hao Jin et al. “Federated Reinforcement Learning with Environment Heterogeneity”. In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, 28–30 Mar 2022, pp. 18–37.

[129] JOhn Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* (2021), pp. 583–589.

[130] Kyle D. Julian and Mykel J. Kochenderfer. “Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning”. In: *Journal of Guidance, Control, and Dynamics* 42.8 (Aug. 2019), pp. 1768–1778. ISSN: 07315090.

[131] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artif. Intell.* 101.1 (May 1998), pp. 99–134. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00023-X.

[132] Peter Kairouz et al. “Advances and open problems in federated learning”. In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210.

[133] Cem Kalkanli and Ayfer Ozgur. “Asymptotic Convergence of Thompson Sampling”. In: *arXiv:2011.03917v1*. 2020.

[134] Cem Kalkanli and Ayfer Ozgur. “Batched Thompson Sampling”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 29984–29994. URL: https://proceedings.neurips.cc/paper/2021/file/fb647ca6672b0930e9d00dc384d8b16f-Paper.pdf.

[135] Ali Taleb Zadeh Kasgari and Walid Saad. "Model-Free Ultra Reliable Low Latency Communication (URLLC): A Deep Reinforcement Learning Framework". In: *IEEE International Conference on Communications (ICC)*. 2019.

[136] Daewoo Kim et al. "Learning to schedule communication in multi-agent reinforcement learning". In: *7th Int. Conf. Learning Repr. (ICLR)*. May 2019.

[137] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd Int. Conf. Learning Repr. (ICLR)*. May 2014.

[138] Jakub Konečnỳ et al. "Federated learning: Strategies for improving communication efficiency". In: *arXiv preprint arXiv:1610.05492* (2016).

[139] Antzela Kosta et al. "The age of information in a discrete time queue: Stationary distribution and non-linear age mean analysis". In: *IEEE J. Sel. Areas Commun.* 39.5 (May 2021), pp. 1352–1364.

[140] Gerhard Kramer and Serap A. Savari. "Communicating Probability Distributions". In: *IEEE Transactions on Information Theory* 53.2 (Jan. 2007), pp. 518–525. DOI: 10.1109/TIT.2006.889015.

[141] Raphail Krichevsky and Victor Trofimov. "The performance of universal encoding". In: *IEEE Transactions on Information Theory* 27.2 (1981), pp. 199–207.

[142] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[143] Akshat Kumar, Shlomo Zilberstein, and Marc Toussaint. "Scalable Multiagent Planning using Probabilistic Inference". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Citeseer. 2011, pp. 2140–2146.

[144] Sampo Kuutti et al. "A survey of deep learning applications to autonomous vehicle control". In: *IEEE Transactions on Intelligent Transportation Systems* 22.2 (Jan. 2020), pp. 712–733.

[145] Branislav Kveton et al. "Matroid bandits: Fast combinatorial optimization with learning". In: *30th Conference Uncertainty in Artificial Intelligence (UAI)* (2014).

[146] Lucy Lai and Samuel J. Gershman. *Policy compression: An information bottleneck in action selection*. Vol. 74. Elsevier Inc., May 2021, pp. 195–232. DOI: 10.1016/bs.plm.2021.02.004.

[147] J.H. Lambert. "Observations Analytiques". In: *Nouveaux m´emoires de l'Acad´emie royale des sciences et belles-lettres* 1 (1770).

[148] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. DOI: 10.1017/9781108571401.

[149] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. "Multi-Agent Cooperation and the Emergence of (Natural) Language". In: *arXiv:1612.07182 [cs]* (Mar. 2017). arXiv: 1612.07182.

[150] M. Lecci et al. "Simplified Ray Tracing for the Millimeter Wave Channel: A Performance Evaluation". In: *Information Theory and Applications Workshop (ITA)*. 2020. DOI: 10.1109/ITA50056.2020.9244950.

[151] Jay Lee, Behrad Bagheri, and Hung-An Kao. "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems". In: *Manufacturing Letters* 3 (Jan. 2015), pp. 18–23. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2014.12.001.

[152] Sergey Levine and U C Berkeley. "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: (). arXiv: arXiv:1805.00909v3. URL: https://arxiv.org/abs/1805.00909.

[153] Ang Li et al. "Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 2021, pp. 42–55.

[154] Ang Li et al. "Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets". In: *arXiv preprint arXiv:2008.03371* (2020).

[155] Cheuk Ting Li and Abbas El Gamal. "Strong Functional Representation Lemma and Applications to Coding Theorems". In: *IEEE Transactions on Information Theory* 64.11 (Nov. 2018), pp. 6967–6978. DOI: 10.1109/TIT.2018.2865570.

[156] Le Liang, Hao Ye, and Geoffrey Ye Li. "Spectrum Sharing in Vehicular Networks Based on Multi-Agent Reinforcement Learning". In: *IEEE Journal on Selected Areas in Communications* 37.10 (Aug. 2019), pp. 2282–2292. DOI: 10.1109/JSAC.2019.2933962.

[157] Sheng Lin et al. "Esmfl: Efficient and secure models for federated learning". In: *arXiv preprint arXiv:2009.01867* (2020).

[158] Xingqin Lin et al. "5G new radio: Unveiling the essentials of the next generation wireless access technology". In: *IEEE Communications Standards Magazine* 3.3 (Sept. 2019), pp. 30–37.

[159] Yujun Lin et al. "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training". In: *International Conference on Learning Representations*. 2018.

[160] Chen-Feng Liu and Mehdi Bennis. "Data-driven predictive scheduling in ultra-reliable low-latency industrial IoT: A generative adversarial network approach". In: *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2020.

[161] Chi Harold Liu et al. "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach". In: *IEEE Journal on Selected Areas in Communications* 36.9 (Aug. 2018), pp. 2059–2070.

[162] Liyuan Liu et al. "On the variance of the adaptive learning rate and beyond". In: *arXiv preprint arXiv:1908.03265* (Aug. 2019).

[163] Tianyu Liu et al. "Measurement-Based Characterization and Modeling for Low-Altitude UAV Air-to-Air Channels". In: *IEEE Access* 7 (2019), pp. 98832–98840.

[164] Xiao Liu, Yuanwei Liu, and Yue Chen. "Reinforcement learning in multiple-UAV networks: Deployment and movement design". In: *IEEE Transactions on Vehicular Technology* 68.8 (June 2019), pp. 8036–8049.

[165] Yan Liu et al. "Analyzing grant-free access for URLLC service". In: *IEEE Journal on Selected Areas in Communications* 39.3 (Mar. 2020), pp. 741–755.

[166] Yang Liu et al. "FedPrune: Personalized and Communication-Efficient Federated Learning on Non-IID Data". In: *International Conference on Neural Information Processing*. Springer. 2021, pp. 430–437.

[167] Dor Livne and Kobi Cohen. "PoPS: Policy Pruning and Shrinking for Deep Reinforcement Learning". In: *IEEE Journal of Selected Topics in Signal Processing* 14.4 (May 2020), pp. 789–801. DOI: 10.1109/JSTSP.2020.2967566.

[168] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[169] M Carmen Lucas-Estañ, Javier Gozalvez, and Miguel Sepulcre. "On the capacity of 5G NR grant-free scheduling with shared radio resources to support ultra-reliable and low-latency communications". In: *Sensors* 19.16 (Aug. 2019), p. 3575.

[170] Federico Mason et al. "Multi-Agent Reinforcement Learning for Pragmatic Communication and Control". In: *arXiv:2302.14399* (Feb. 2023).

[171] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[172] H Brendan McMahan et al. "Learning differentially private recurrent language models". In: *arXiv preprint arXiv:1710.06963* (2017).

[173] Ilya Mironov. "Rényi differential privacy". In: *IEEE 30th computer security foundations symposium (CSF)*. IEEE. 2017, pp. 263–275.

[174] Nicole Mitchell et al. "Optimizing the communication-accuracy trade-off in federated learning with rate-distortion theory". In: *arXiv preprint arXiv:2201.02664* (2022).

[175] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.

[176] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. June 2016, pp. 1928–1937.

[177] Amirkeivan Mohtashami, Martin Jaggi, and Sebastian Stich. "Masked Training of Neural Networks with Partial Gradients". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 5876–5890.

[178] Hamid Mozaffari, Virat Shejwalkar, and Amir Houmansadr. "FRL: Federated Rank Learning". In: *arXiv preprint arXiv:2110.04350* (2021).

[179] Muhammad Tahir Munir et al. "FedPrune: Towards Inclusive Federated Learning". In: *arXiv preprint arXiv:2110.14205* (2021).

[180] Ranjit Nair et al. "Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs". In: *19th Conference on Artificial Intelligence*. Vol. 5. AAAI. July 2005, pp. 133–139.

[181] Minoru Nakagami. "The m-Distribution–A General Formula of Intensity Distribution of Rapid Fading". In: *Statistical Methods in Radio Wave Propagation* (1960). Ed. by W.C. HOFFMAN, pp. 3–36.

[182] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*. Springer, Apr. 2016.

[183] OpenAI. "GPT-4 Technical Report". In: *arXiv:2303.08774* (2023).

[184] Emre Ozfatura, Kerem Ozfatura, and Deniz Gündüz. "Time-correlated sparsification for communication-efficient federated learning". In: *IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2021, pp. 461–466.

[185] Stefan Parkvall, Erik Dahlman, and Johan Sköld. *5G NR: The Next Generation Wireless Access Technology*. Academic Press, 2018.

[186] Francesco Pase, Marco Giordani, and Michele Zorzi. "On the Convergence Time of Federated Learning Over Wireless Networks Under Imperfect CSI". In: *IEEE International Conference on Communications Workshops (ICC)*. 2020.

[187] Francesco Pase, Deniz Gündüz, and Michele Zorzi. "Rate-Constrained Remote Contextual Bandits". In: *IEEE Journal on Selected Areas in Information Theory* (Dec. 2022).

[188] Francesco Pase, Deniz Gündüz, and Michele Zorzi. "Remote Contextual Bandits". In: *IEEE International Symposium on Information Theory (ISIT)* (2022).

[189] Francesco Pase, Deniz Gündüz, and Michele Zorzi. "Remote Contextual Bandits". In: *2022 IEEE International Symposium on Information Theory (ISIT)*. 2022.

[190] Francesco Pase et al. "A Distributed Neural Linear Thompson Sampling Framework to Achieve URLLC in Industrial IoT". In: ***under review for*** *the IEEE Transaction on Wireless Communications* (2024).

[191] Francesco Pase et al. "Adaptive Compression in Federated Learning via Side Information". In: ***under review for*** *the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2024.

[194] Francesco Pase et al. "Distributed Resource Allocation for URLLC in IIoT Scenarios: A Multi-Armed Bandit Approach". In: *IEEE GLOBECOM Workshops (GC Wkshps)* (2022).

[197] Francesco Pase et al. "Leveraging Side Information for Communication-Efficient Federated Learning". In: *International Conference on Learning Representation (ICML): Workshop on Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities* (2023).

[198] Francesco Pase et al. "Semantic Communication of Learnable Concepts". In: *IEEE International Symposium on Information Theory* (2023).

[200] Francesco Pase et al. "Sparse Random Networks for Communication-Efficient Federated Learning". In: *The Eleventh International Conference on Learning Representations*. 2023.

[201] Ankit Pensia et al. "Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2599–2610.

[202] My Phan, Yasin Abbasi Yadkori, and Justin Domke. "Thompson Sampling and Approximate Inference". In: *Advances in Neural Information Processing Systems*. 2019.

[203] Vincent Plassier et al. "DG-LMC: a turn-key and scalable synchronous distributed MCMC algorithm via Langevin Monte Carlo within Gibbs". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8577–8587.

[204] M. Polese, M. Giordani, and M. Zorzi. "3GPP NR: the cellular standard for 5G networks". In: *5G-ITALY White Book* (2019).

[205] Petar Popovski et al. "Semantic-effectiveness filtering and control for post-5G wireless connectivity". In: *J. Indian Inst. Sci.* 100.2 (Apr. 2020), pp. 435–443.

[206] Zeyu Qin et al. "Multi-Agent Reinforcement Learning Aided Computation Offloading in Aerial Computing for the Internet-of-Things". In: *IEEE Transactions on Services Computing* 16.3 (2023), pp. 1976–1986. DOI: 10.1109/TSC.2022.3190562.

[207] Vivek Ramanujan et al. "What's hidden in a randomly weighted neural network?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11893–11902.

[208] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8821–8831. URL: https://proceedings.mlr.press/v139/ramesh21a.html.

[209] T. S. Rappaport et al. “Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!” In: *IEEE Access* 1 (2013), pp. 335–349. DOI: 10.1109/ACCESS.2013.2260813.

[210] Carlos Riquelme, George Tucker, and Jasper Snoek. “Deep Bayesian Bandits Showdown”. In: *International Conference on Learning Representations (ICLR)* (2018).

[211] Jorma Rissanen and Glen G Langdon. “Arithmetic coding”. In: *IBM Journal of research and development* 23.2 (1979), pp. 149–162.

[212] D. J. Russo et al. In: *Found. Trends Mach. Learn.* 11.1 (2018). ISSN: 1935-8237.

[213] Daniel Russo. “Simple Bayesian Algorithms for Best Arm Identification”. In: *29th Annual Conference on Learning Theory*. Ed. by Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir. Vol. 49. Proceedings of Machine Learning Research. Columbia University, New York, New York, USA: PMLR, 23–26 Jun 2016, pp. 1417–1418. URL: https://proceedings.mlr.press/v49/russo16.html.

[214] Daniel Russo and Benjamin Van Roy. “Learning to optimize via posterior sampling”. In: *Mathematics of Operations Research* 39.4 (Apr. 2014), pp. 1221–1243. DOI: 10.1287/moor.2014.0650.

[215] M. K. Samimi et al. “28 GHz Millimeter-Wave Ultrawideband Small-Scale Fading Models in Wireless Channels”. In: *IEEE 83rd Vehicular Technology Conference* (2016). DOI: 10.1109/VTCSpring.2016.7503970.

[216] Mahadev Satyanarayanan. “The Emergence of Edge Computing”. In: *Computer* 50.1 (Jan. 2017), pp. 30–39. DOI: 10.1109/MC.2017.9.

[217] Stefano Savazzi et al. “Opportunities of Federated Learning in Connected, Cooperative and Automated Industrial Systems”. In: *arXiv preprint arXiv:2101.03367* (2021).

[218] Sejin Seo et al. “Communication-efficient and personalized federated lottery ticket learning”. In: *IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. 2021, pp. 581–585.

[219] Abhin Shah et al. “Optimal compression of locally differentially private mechanisms”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 7680–7723.

[220] Reza Shakeri et al. “Design Challenges of Multi-UAV Systems in Cyber-Physical Applications: A Comprehensive Survey and Future Directions”. In: *IEEE Communications Surveys and Tutorials* 21.4 (Apr. 2019), pp. 3340–3385. ISSN: 1553877X.

[221] Hazim Shakhatreh et al. “Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges”. In: *IEEE Access* 7 (2019), pp. 48572–48634. ISSN: 21693536.

[222] Claude E. Shannon and Warren Weaver. “The Mathematical Theory of Communication.” In: *University of Illinois Press* (1949).

[223] Jiawei Shao, Yuyi Mao, and Jun Zhang. “Learning task-oriented communication for edge inference: An information bottleneck approach”. In: *IEEE J. Sel. Areas Commun.* 40.1 (Jan. 2022), pp. 197–211.

[224] Yulin Shao, Qi Cao, and Deniz Gündüz. “A Theory of Semantic Communication”. In: *arXiv:2212.01485* (Dec. 2022).

[225] Chengshuai Shi and Cong Shen. “Federated Multi-Armed Bandits”. In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)* (2021).

[226] Aleksandrs Slivkins. "Introduction to Multi-Armed Bandits". In: *Foundations and Trends® in Machine Learning* 12 (2019). URL: http://dx.doi.org/10.1561/2200000068.

[227] Edward J Sondik. "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs". In: *Oper. Res.* 26.2 (Apr. 1978), pp. 282–304.

[228] Mingjun Song and Sanguthevar Rajasekaran. "Fast k-Means Algorithms with Constant Approximation". In: *Algorithms and Computation*. Springer Berlin Heidelberg, 2005, pp. 1029–1038.

[229] Photios A Stavrou and Marios Kountouris. "A rate distortion approach to goal-oriented communication". In: *Int. Symp. Inform. Theory (ISIT)*. IEEE. June 2022, pp. 590–595.

[230] Thomas Steinke and Lydia Zakynthinou. "Reasoning About Generalization via Conditional Mutual Information". In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. PMLR, July 2020, pp. 3437–3452.

[231] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. "Learning multiagent communication with backpropagation". In: *Proc. of 30th Int'l Conf. on Neural Information Proc. Systems*. NIPS'16. Red Hook, NY, 2016, pp. 2252–2260.

[232] Ananda Theertha Suresh et al. "Distributed mean estimation with limited communication". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3329–3337.

[233] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, Nov. 2018.

[234] Pietro Talli et al. "Semantic and Effective Communication for Remote Control Tasks with Dynamic Feature Compression". In: *16th Int. Worksh. Wireless Sensing Actuating Rob. Networks (INFOCOM WiSARN)*. IEEE. May 2023.

[235] Lucas Theis and Noureldin Y Ahmed. "Algorithms for the Communication of Samples". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. PMLR, July 2022, pp. 21308–21328.

[236] Lucas Theis and Noureldin Y Ahmed. "Algorithms for the communication of samples". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 21308–21328.

[237] William R. Thompson. "On the Theory of Apportionment". In: *American Journal of Mathematics* 57.2 (Apr. 1935), pp. 450–456.

[238] Naftali Tishby, Fernando C. Pereira, and William Bialek. *The information bottleneck method*. 2000. DOI: 10.48550/ARXIV.PHYSICS/0004057. URL: https://arxiv.org/abs/physics/0004057.

[239] Hugo Touvron et al. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv:2302.13971* (2023).

[240] Aleksei Triastcyn, Matthias Reisser, and Christos Louizos. "DP-REC: Private & communication-efficient federated learning". In: *arXiv preprint arXiv:2111.05454* (2021).

[241] Tze-Yang Tung et al. "Effective Communications: A Joint Learning and Communication Framework for Multi-Agent Reinforcement Learning Over Noisy Channels". In: *IEEE Journal on Selected Areas in Communications* 39.8 (June 2021), pp. 2590–2603. DOI: 10.1109/JSAC.2021.3087248.

[242] Elif Uysal et al. "Semantic communications in networked systems: A data significance perspective". In: *IEEE Network* 36.4 (Oct. 2022), pp. 233–240.

[243] Anish K Vallapuram et al. "HideNseek: Federated Lottery Ticket via Server-side Pruning and Sign Supermask". In: *arXiv preprint arXiv:2206.04385* (2022).

[244] Aaron Van Den Oord, Oriol Vinyals, et al. "Neural discrete representation learning". In: *31st Int. Conf. Neural Inf. Proc. Sys. (NeurIPS)*. Dec. 2017.

[245] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with Double Q-learning". In: *30th Conference on Artificial Intelligence*. AAAI. Mar. 2016.

[246] Shay Vargaftik et al. "DRIVE: one-bit distributed mean estimation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 362–377.

[247] Shay Vargaftik et al. "Eden: Communication-efficient and robust distributed mean estimation for federated learning". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 21984–22014.

[249] Federico Venturini et al. "Distributed Reinforcement Learning for Flexible UAV Swarm Control with Transfer Learning Capabilities". In: *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet)*. 2020.

[250] Timothy Verstraeten, Eugenio Bargiacchi, and Pieter .J.K. et al. Libin. "Multi-Agent Thompson Sampling for Bandit Applications with Sparse Neighbourhood Structures". In: *35th Conference on Neural Information Processing Systems (NeurIPS)* (2020).

[251] Oriol Vinyals et al. "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II". In: *https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii* (2019).

[252] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. "PowerSGD: Practical low-rank gradient compression for distributed optimization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[253] Maxime Vono et al. "QLSD: Quantised Langevin stochastic dynamics for Bayesian federated learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 6459–6500.

[254] M. Wadu, S. Samarakoon, and M. Bennis. "Federated Learning under Channel Uncertainty: Joint Client Scheduling and Resource Allocation". In: *IEEE Wireless Communications and Networking Conference*. 2020. DOI: 10.1109/WCNC45663.2020.9120649.

[255] Shaohua Wan, Zonghua Gu, and Qiang Ni. "Cognitive computing and wireless communications on the edge for healthcare service robots". In: *Comput. Commun.* 149 (Jan. 2020), pp. 99–106.

[256] Hongyi Wang et al. "Atomo: Communication-efficient learning via atomic sparsification". In: *Advances in Neural Information Processing Systems* 31 (2018).

[257] Mowei Wang et al. "Machine learning for networking: Workflow, advances and opportunities". In: *IEEE Network* 32.2 (Mar. 2017), pp. 92–99.

[258] Siwei Wang and Wei Chen. "Thompson sampling for combinatorial semi-bandits". In: *International Conference on Machine Learning (ICML)* (2018).

[259] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. "Subsampled Renyi Differential Privacy and Analytical Moments Accountant". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 1226–1235.

[260] Zijing Wang, Mihai-Alin Badiu, and Justin P Coon. "A framework for characterizing the value of information in hidden Markov models". In: *IEEE T. Inform. Theory* 68.8 (Aug. 2022), pp. 5203–5216.

[261] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 681–688.

[262] Wei Wen et al. "Terngrad: Ternary gradients to reduce communication in distributed deep learning". In: *Advances in Neural Information Processing Systems* 30 (2017).

[263] Martin Wollschlaeger, Thilo Sauter, and Juergen Jasperneite. "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0". In: *IEEE Industrial Electronics Magazine* 11.1 (Mar. 2017), pp. 17–27.

[264] Chunxue Wu et al. "UAV autonomous target search based on deep reinforcement learning in complex disaster scene". In: *IEEE Access* 7 (2019), pp. 117227–117245.

[265] Huiqiang Xie et al. "Deep Learning Enabled Semantic Communication Systems". In: *IEEE T. Signal Proc.* 69 (Apr. 2021), pp. 2663–2675.

[266] Aolin Xu and Maxim Raginsky. "Information-theoretic analysis of generalization capability of learning algorithms". In: *Proceedings of the 31th International Conference on Neural Information Processing Systems*. Long Beach, CA, USA, 2017.

[267] Lei Xu et al. "Information Security in Big Data: Privacy and Data Mining". In: *IEEE Access* 2 (Oct. 2014), pp. 1149–1176. DOI: 10.1109/ACCESS.2014.2362522.

[268] Helin Yang, Xianzhong Xie, and Michel Kadoch. "Intelligent Resource Management Based on Reinforcement Learning for Ultra-Reliable and Low-Latency IoV Communication Networks". In: *IEEE Transactions on Vehicular Technology* 68.5 (Jan. 2019), pp. 4157–4169. DOI: 10.1109/TVT.2018.2890686.

[269] Peihao Yang, Linghe Kong, and Guihai Chen. "Spectrum Sharing for 5G/6G URLLC: Research Frontiers and Standards". In: *IEEE Communications Standards Magazine* 5.2 (Apr. 2021), pp. 120–125. DOI: 10.1109/MCOMSTD.001.2000054.

[270] Zhaohui Yang et al. "Delay Minimization for Federated Learning over Wireless Communication Networks". In: *37 th International Conference on Machine Learning* (2020).

[271] LeCun Yann and Cortes Corinna. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[272] R. D. Yates et al. "Age of Information: An Introduction and Survey". In: *IEEE J. Sel. Areas Commun.* 39.5 (Mar. 2021), pp. 1183–1210.

[273] Sixing Yu et al. "Adaptive dynamic pruning for non-iid federated learning". In: *arXiv preprint arXiv:2106.06921* (2021).

[274] Riccardo Zanol, Federico Chiariotti, and Andrea Zanella. "Drone mapping through multi-agent reinforcement learning". In: *Wireless Communications and Networking Conference (WCNC)*. IEEE, Apr. 2019, pp. 1–7. ISBN: 9781538676462.

[275] Chongjie Zhang and Victor R Lesser. "Coordinated Multi-Agent Reinforcement Learning in Networked Distributed POMDPs". In: *AAAI*. Aug. 2011.

[276] Hongjie Zhang, Zhuocheng He, and Jing Li. "Accelerating the Deep Reinforcement Learning with Neural Network Compression". In: *2019 International Joint Conference on Neural Networks (IJCNN)* (2019). DOI: 10.1109/IJCNN.2019.8852451.

[277] Senbai Zhang et al. "Multi-agent Reinforcement Learning-Based Orbital Edge Offloading in SAGIN Supporting Internet of Remote Things". In: *IEEE Internet of Things Journal* (2023), pp. 1–1. DOI: 10.1109/JIOT.2023.3287737.

[278] Xi Zheng, Sheng Zhou, and Zhisheng Niu. "Urgency of Information for Context-Aware Timely Status Updates in Remote Control Systems". In: *IEEE T. Wireless Commun.* 19.11 (July 2020), pp. 7237–7250.

[279] Hattie Zhou et al. "Deconstructing lottery tickets: Zeros, signs, and the supermask". In: *Advances in neural information processing systems* 32 (2019).

[280] T. Zugno et al. "Toward Standardization of Millimeter-Wave Vehicle-to-Vehicle Networks: Open Challenges and Performance Evaluation". In: *IEEE Communications Magazine* 58.9 (Sept. 2020), pp. 79–85. DOI: 10.1109/MCOM.001.2000041.

List of publications

Journals

[187] Francesco Pase, Deniz Gündüz, and Michele Zorzi. "Rate-Constrained Remote Contextual Bandits". In: *IEEE Journal on Selected Areas in Information Theory* (Dec. 2022).

[190] Francesco Pase et al. "A Distributed Neural Linear Thompson Sampling Framework to Achieve URLLC in Industrial IoT". In: ***under review for*** *the IEEE Transaction on Wireless Communications* (2024).

[248] Federico Venturini et al. "Distributed Reinforcement Learning for Flexible and Efficient UAV Swarm Control". In: *IEEE Transactions on Cognitive Communications and Networking* 7.3 (2021), pp. 955–969. DOI: 10.1109/TCCN.2021.3063170.

Conferences

[56] S. Cavallero et al. "A New Scheduler for URLLC in 5G NR IIoT Networks with Spatio-Temporal Traffic Correlations". In: *IEEE International Conference on Communications (ICC)*. 2023.

[74] G. Cuozzo et al. "Enabling URLLC in 5G NR IIoT Networks: A Full-Stack End-to-End Analysis". In: *Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 2022.

[186] Francesco Pase, Marco Giordani, and Michele Zorzi. "On the Convergence Time of Federated Learning Over Wireless Networks Under Imperfect CSI". In: *IEEE International Conference on Communications Workshops (ICC)*. 2020.

[188] Francesco Pase, Deniz Gündüz, and Michele Zorzi. "Remote Contextual Bandits". In: *IEEE International Symposium on Information Theory (ISIT)* (2022).

[191] Francesco Pase et al. "Adaptive Compression in Federated Learning via Side Information". In: ***under review for*** *the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2024.

[194] Francesco Pase et al. "Distributed Resource Allocation for URLLC in IIoT Scenarios: A Multi-Armed Bandit Approach". In: *IEEE GLOBECOM Workshops (GC Wkshps)* (2022).

[195] Francesco Pase et al. "Efficient Federated Random Subnetwork Training". In: *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*. 2022.

[197] Francesco Pase et al. "Leveraging Side Information for Communication-Efficient Federated Learning". In: *International Conference on Learning Representation (ICML): Workshop on Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities* (2023).

[198] Francesco Pase et al. "Semantic Communication of Learnable Concepts". In: *IEEE International Symposium on Information Theory* (2023).

[200] Francesco Pase et al. "Sparse Random Networks for Communication-Efficient Federated Learning". In: *The Eleventh International Conference on Learning Representations*. 2023.

[234] Pietro Talli et al. “Semantic and Effective Communication for Remote Control Tasks with Dynamic Feature Compression”. In: *16th Int. Worksh. Wireless Sensing Actuating Rob. Networks (INFOCOM WiSARN)*. IEEE. May 2023.

[249] Federico Venturini et al. “Distributed Reinforcement Learning for Flexible UAV Swarm Control with Transfer Learning Capabilities”. In: *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet)*. 2020.

Patents

[192] Francesco Pase et al. *Data Hetereougenity in Federated Learning*. **Under filing** with Nokia Bell Labs.

[193] Francesco Pase et al. *Devices and methods for AI-assisted URLLC*. **Filed** with Huawei Technology.

[196] Francesco Pase et al. *Improving Foundation Model Training Through Parameter-Efficient Federated Learning*. **Under filing** with Nokia Bell Labs.

Book Chapter

[199] Francesco Pase et al. *Smart Data Gathering for Network Optimization*. CNIT Technical Report-07, 2001.

Acknowledgments

This manuscript enclosed the research journey of my PhD, which has been a tremendously beautiful, inspiring and vibrant experience, and has provided to me the necessary tools to address new research questions. These years have not been always easy, ups and downs were present in almost every week, specially when things on which I spent many hours did not work out initially, and when I started questioning my future plans. However the experience I have accumulated, both on the personal and working sphere, is now helping me realizing how to steer my years ahead, and it is probably one of the things I'm mostly proud of. Needless to say, all my achievements were not possible without the people I shared my life with, who I'm going to thank in what will probably be the most read section of this dissertation.

As anticipated, this thesis goes to my parents, Gianmaria and Carla, who always supported me in all my adventures, and whose sacrifices allowed me to pursue my passions. This special mention is also enlarged to my older brothers, Giovanni and Alessandro, from whom I learned a lot during my life. Thank you all.

First and foremost, I want to thank my supervisor, Michele Zorzi. You have always believed in my ideas and plans, even after I dropped out from the first PhD in Lausanne, welcoming me in your group. You made my entire PhD experience smooth, providing me all the necessary to pursue my research. Even though due to the pandemic restrictions and our commitments we did not spend so much time in the lab together, the always stimulating conversations we had in your office or on a phone call shaped my thinking, your points have always been precious fruits for my research, and I'm extremely thankful for that. I'm very grateful to Deniz Gunduz, who introduced my into the complex but incredibly fascinating world of information theory, and invited me to spend part of my journey at the Imperial College London, which revealed to be one of the most beautiful periods in my entire life. It has been a real honor to do research with you, and I really hope it will happen again in the future. To my other amazing mentors Dimitrios, Mohammad and Soumayajit, who made my internship at Nokia Bell Labs a truly inspiring experience thanks to their patience, suggestions, and valuable insights, it has been a real pleasure to share my summer in Cambridge with you (sorry Sumo if Samsung already patented everything). To Fahim Kawsar, for making my internship possible, and for believing in me to participate to the successful student competition: discussing my internship project and preparing the final presentation with you has been profoundly helpful.

To the SIGNET guys, who have been like a family during the whole PhD: Jacopo, Federico M., Paolo, Mattia, Matteo, Tommaso, Andrea, Sara, Matteo P., Alessandro T., Francesca, Enver, Martina, Anay, Salman, Roberto, Filippo, Giovanni, Riccardo, Silvia, Marco C., Pietro T., Alberto Z., Zaman. A special gratitude to Marco Giordani and Federico Chiariotti, the seniors who gave me important lessons for my research, and to the fabulous Lab 219, Alberto, Laura, Francesco, Amir, Neda, Aria, Ahmed and Leonardo, I'll never forget the days spent in the lab with you

(and thanks for bearing my strange working habits). To the other professors of the Lab: Andrea Zanella, Michele Rossi, Leonardo Badia, Federica Battisti and Lorenzo Vangelista, for considering every one of us as a peer, and for your valuable lectures during my undergraduate and graduate classes. To the rest of the floor Daniele, Matteo C., Federico L., Francesco B., Adriano, Marco T., Giulia, Donald, Elena. A special thank to Umberto and Mattia, for teaching me the fundamental PhD tricks, those days in the islands were unforgettable. A special thank goes also to my dear and closest University's friends Andrea, Luca, Francesco, Alessandro, Laura, Giulia and Alice.

To the Imperial College friends, it was such a pleasure being part of your group. A special mention goes to Roy, Szymon, Mohammad, Ece and Rui, who beacame more than Lab mates. Thanks Roy for your wise insights (specially regarding TV series), and for the post-work conversations with a beer at the Union; and Szymon, for our crazy research discussions walking to the Starbucks during the afternoons, and for our clubbing experience in Taipei, that was missing. To my colleagues at Nokia Bell Labs Simon, Aashish, Julia, Irtaza, Talia, Ila, Haerang, Arthur, Adiba, it was inspiring to know each of you, and to hear the stories such interesting and stimulating people coming from all over the world. To the amazing team at Bell Labs, I was admired by the things your were able to achieve working as a team, and by how you welcomed us to be part of your days as in a real family. A special note for Andrea, together with whom I was unbeatable at table football, and to the other amazing finalists Ryo and Alessandro. To Mirco, my flatmate in Cambridge, who is now a friend. We shared together our temporary polish family Pawel and Justyna, who I also want to thank for their warm hospitality. To my friend Matteo C., who welcomed me with the best Amatriciana in the city. To my wonderful collaborator Berivan, thanks for all the hard work we did together, for the stimulating research discussions, and for the time we spent together at the conferences. To her supervisor Tsachy Weissman, for his precious feedback on our work.

To my second family Marco P., Alberto, Marco O., Michele, Tobia, Ruben, for all your support, and because you're one of the reasons why it's always a joy to come back home. To my other home's friends, who quickly became special life mates Riccardo, Alessandro, Matteo, Marco P., Daniele, Francesca, Camilla, Yasmine, Greta, Valentina, Giorgia. To my high school friends, specially those who I cannot stop spending time with like Giulio, Tobia, Marco, Giulia, Matteo. To my amazing mates Gianmarco, Cristiana, and closest friends in Padova Sabrina, Christian, Valentina, Andrea, Alissa, Simone. A special thank to Francesco, who believed in me offering my next adventure at NEWTWEN, it's a honor to be part of such an amazing team, and I'm very excited by what we can do next. To Tito & Dani, lifetime and very important friends, for their never-missing support, and for having fed in me the passion for technology.

To my friends Edoardo (because one is not just a number), Giacomo P.(my clubbing mate) and the rest of the italian London's crew Giacomo T., Ruggero, Filippo, Stefano, Amedeo and Giovanni, who contributed making my London's time unforgettable. Special gratitude goes also to all the other people who enriched my time over the past 3 years, Bianca, Reza, Sara, Saket, Ettore, Tommy, Lodovica, Giada, Stefano, Veronica, Nitish, Mine, Selim, Mikolaj.

Thank you all for contributing to my proudest achievement.