Distributed Learning in Wireless Networks: Recent Progress and Future Challenges

Mingzhe Chen, *Member, IEEE*, Deniz Gündüz, *Senior Member, IEEE*, Kaibin Huang, *Fellow, IEEE*, Walid Saad, *Fellow, IEEE*, Mehdi Bennis, *Fellow, IEEE*, Aneta Vulgarakis Feljan, *Member, IEEE*, and H. Vincent Poor, *Life Fellow, IEEE*,

***Abstract*—The next-generation of wireless networks will enable many machine learning (ML) tools and applications to efficiently analyze various types of data collected by edge devices for inference, autonomy, and decision making purposes. However, due to resource constraints, delay limitations, and privacy challenges, edge devices cannot offload their entire collected datasets to a cloud server for centrally training their ML models or inference purposes. To overcome these challenges, distributed learning and inference techniques have been proposed as a means to enable edge devices to collaboratively train ML models without raw data exchanges, thus reducing the communication overhead and latency as well as improving data privacy. However, deploying distributed learning over wireless networks faces several challenges including the uncertain wireless environment (e.g., dynamic channel and interference), limited wireless resources (e.g., transmit power and radio spectrum), and hardware resources (e.g., computational power). This paper provides a comprehensive study of how distributed learning can be efficiently and effectively deployed over wireless edge networks. We present a detailed overview of several emerging distributed learning paradigms, including federated learning, federated distillation, distributed inference, and multi-agent reinforcement learning. For each learning framework, we first introduce the motivation for deploying it over wireless networks. Then, we present a detailed literature review on the use of communication techniques for its efficient deployment. We then introduce an illustrative example to show how to optimize wireless networks to improve its performance. Finally, we introduce future research opportunities. In a nutshell, this paper provides a holistic set of guidelines on how to deploy a broad range of distributed learning frameworks over real-world wireless communication networks.**

***Index Terms*—Distributed learning, wireless edge networks, federated learning, federated distillation, distributed inference, multi-agent reinforcement learning.**

M. Chen and H. V. Poor are with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ, 08544, USA, Emails: mingzhec@princeton.edu, poor@princeton.edu.

D. Gündüz is with the Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK, Email: d.gunduz@imperial.ac.uk.

K. Huang is with the Department of Electrical & Electronic Engineering, The University of Hong Kong, Hong Kong, Email: huangkb@eee.hku.hk.

W. Saad is with the Wireless@VT, Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24060, USA, Email: walids@vt.edu.

M. Bennis is with the Department of Communications Engineering, University of Oulu, FI-90014, Oulu Finland, Email: mehdi.bennis@oulu.fi.

A. V. Feljan is with Ericsson Research, Stockholm, Sweden, Email: aneta.vulgarakis@ericsson.com.

The work of Mingzhe Chen and H. Vincent Poor was supported by the U.S. National Science Foundation under Grant CCF-1908308. The work of Deniz Gündüz was supported in part by the European Research Council (ERC) through Starting Grant BEACON (no. 677854) and in part by the UK EPSRC (grant no. EP/T023600/1) within the CHIST-ERA program. The work of Walid Saad was supported by the Office of Naval Research (ONR) under MURI Grant N00014-19-1-2621.

I. Introduction

A. Motivation of Distributed Learning

Over the past five years, the field of machine learning (ML) has witnessed a major shift from the so-called "big data" paradigm, in which large volumes of data are collected and processed at a central cloud, towards a "small data" paradigm [1]–[3], in which a set of agents or devices distributively process their data at the edge of a mobile network. The main motivation of this paradigm shift is to allow edge devices to rapidly access real-time data for fast ML model training. This in turn endows on the devices human-like intelligence to respond to real-time events [4]–[9].

This paradigm shift is driven by two trends in the evolution of computing. First, as computer chips become cheaper, computers are built into tens of billions of devices. They are connected to form Internet-of-Things (IoT) networks, which provide platforms for executing large-scale tasks but also generate very large amounts of useful data. Second, the spread of computing from the cloud towards the network edge enables the deployment of ML algorithms in the proximity of edge devices to distill their collected data into intelligence. This paradigm shift means that the classical centralized ML approach requiring large training datasets is no longer dominant. There is a growing need for novel *distributed learning* solutions that can leverage rich distributed data and computation resources at the edge without the need for transporting data across the network. The new framework of distributed learning finds a wide range of applications especially those related to IoT such as connected autonomy (e.g., connected vehicles or drones). In such systems, under the constraint of data privacy, devices have to find an intelligent way to cooperate in training an ML model by overcoming their local-data scarcity. In such scenarios, the direct exchange of raw data is undesirable due to privacy concerns, or, in some cases, even infeasible due to communication and computing constraints.

B. Challenges of Deploying Distributed Learning

Realizing a successful evolution towards distributed learning requires overcoming several key challenges. The first is to find methods for distributed learning without raw-data sharing. This gives rise to an interesting trade-off between privacy and learning accuracy as regulated by the level of information exchange among distributed agents. The second challenge stems from the fact that one common denominator of all such distributed systems is the need to perform training and inference of an ML model over wireless links as devices are usually connected using a cellular network or a wireless

local area network (WLAN). As such, the characteristics of wireless propagation – interference, noise, and fading – will now introduce new impediments to the learning process [10]. For example, in [11], it has been shown that bit errors and communication delay can significantly affect the convergence and accuracy of distributed learning. Furthermore, it is shown in [12] that the wireless network architecture can also have a significant effect on the convergence speed of a learning algorithm. Hence, the deployment of distributed learning in wireless networks entails a need for accounting for wireless factors in the design. A third, related challenge, is the fact that distributed learning requires many rounds of exchanging high-dimensional ML model updates or model parameters between parameter servers and devices. However, the radio spectrum is a scarce resource. To resolve the conflict calls for the design of communication-efficient techniques for distributed learning. The fourth challenge pertains to computing. Distributed learning requires efficient ways to perform distributed computation, both over-the-air [13]–[15] and off-the-air. The delay and efficiency associated with distributed ML and distributed computing over large scale wireless networks will directly impact the learning performance. The final challenge is that efficient distributed learning over wireless systems requires new distributed optimization frameworks that enable multiple agents to collaboratively solve complex optimization problems in a distributed way.

Research efforts aiming at addressing these challenges have led to the emergence of many important distributed learning frameworks in the past few years. Perhaps the most widely-studied one is *federated learning (FL)* [16] which enables a group of agents to collaboratively execute a common learning task by exchanging only their local model parameters instead of raw data. Thereby, FL helps preserve data privacy while achieving high learning accuracy. Following the seminal work in [16], a broad range of FL techniques have been developed to tackle individual challenges among those mentioned earlier. At the same time, in the direction of distributed optimization, the framework of multi-agent reinforcement learning (MARL) [17] is gaining rapidly growing popularity. By combining the concept of RL with deep neural networks (DNNs) as well as distributed multi-agent control, one can enable a group of agents to solve a set of distributed optimization problems, without the need to rely on global information or without excessive exchange of data. MARL itself faces many unique challenges including the guarantee of convergence, optimality, and the support of real-time operations. Both FL and MARL will have to operate over large-scale wireless systems. This subjects them to the wireless-related issues as described earlier.

C. Potential Techniques for Deploying Distributed Learning

To achieve very high performance in different dimensions, 6G will feature the integrated design of sensing, communication, computing, and control. In the context of distributed learning, the objective of 6G design is no longer rate maximization but to accelerate the training of ML models using distributed data [18]–[23]. This requires new algorithms and techniques for integrated communication and learning. A first approach is compression and sparsification. Compression techniques aim at using fewer bits to quantize each ML model parameter [24]–[26] while the objective of sparsification techniques is to transform high-dimensional ML model updates to their sparse representations by pruning some relatively unimportant elements [27]–[29]. As a result, they decrease the size of the ML model parameters or updates exchanged among devices to reduce the communication overhead. A second approach is radio resource management [30], [31] which enables wireless networks to efficiently use the limited resources such as spectrum, transmit power, and computational capabilities to complete the distributed learning process. Over-the-air computation (OAC or AirComp) [32] is a third approach that provides the needed scalability for multiple-access in distributed learning to the participation of many edge devices which is crucial for satisfactory learning performance. A fourth approach is to develop novel training methods that jointly consider distributed learning parameters, wireless network dynamics (e.g., wireless channel conditions), and wireless network topologies [33] (e.g., locations and mobility patterns of wireless devices).

D. Outline

In this article, we introduce the challenges, solutions, and research opportunities associated with distributed learning over wireless networks. The main focus of this article is the widely-studied FL framework for distributed learning. In this context, we first provide a detailed overview of federated averaging, federated multi-task learning, and model agnostic meta learning based FL and summarize their drawbacks and usage. Then, we explore the possibility of performing joint learning and communications when FL is deployed in wireless networks. We first introduce four important performance metrics to quantify the FL performance over wireless networks and analyze how wireless factors affect these metrics. Next, we discuss novel approaches ranging from compression and sparsification, wireless resource management, FL training method design, and AirComp, to optimize the FL performance metrics, while taking into account the need for communication-efficient learning and effective distributed computing. The discussion of each approach spans literature review, design example, and future research opportunities. Next, we delve further into communication-efficient FL and present the concept of federated distillation (FD) and its ramifications. Subsequently, we discuss the use of distributed learning for inference. Finally, MARL is introduced as a powerful tool for distributed learning and optimization.

The rest of this paper is organized as follows. In Section II, we introduce communication efficient FL. Section III presents AirComp based FL. In Section IV, we present FD. Section V introduces distributed inference over wireless networks. In Section VI, we introduce MARL over wireless networks. Finally, conclusions are drawn in Section VII.

II. Communication Efficient Federated Learning

We first introduce the preliminaries of FL. In particular, we introduce the federated averaging and personalized FL

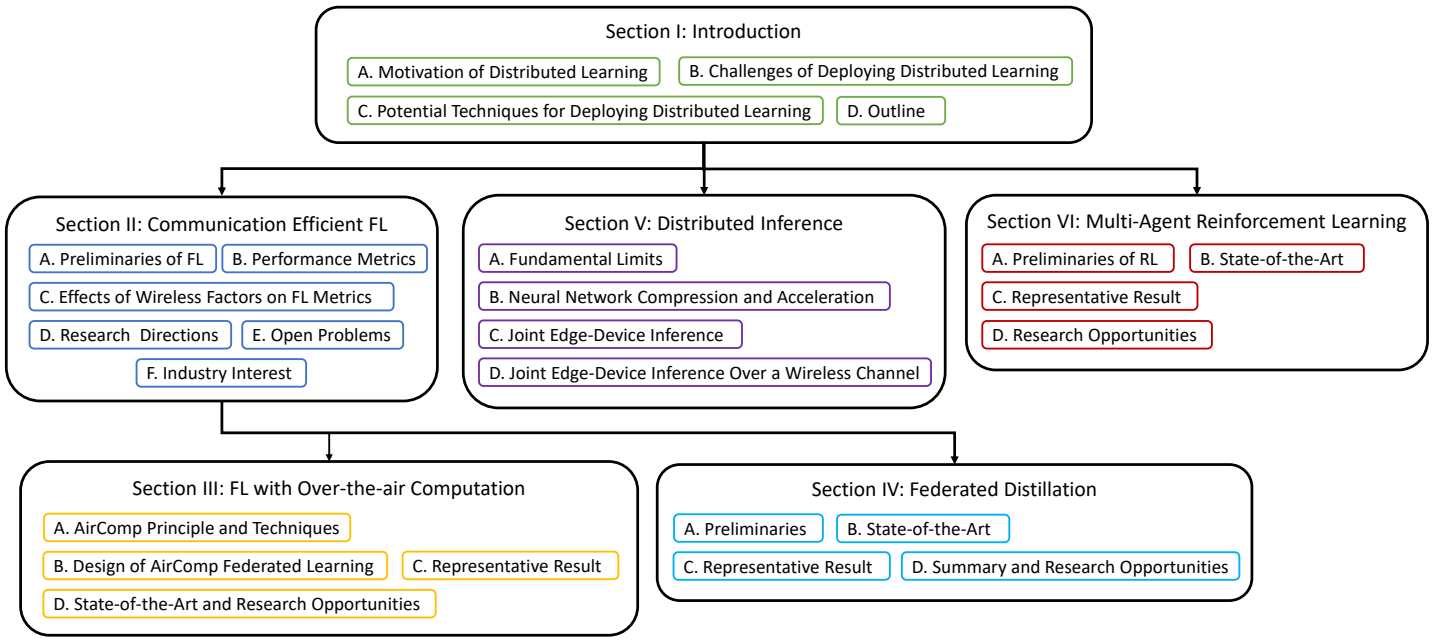


Fig. 1. Organization.

algorithms. Then, we introduce four important performance metrics to quantify the performance of FL over wireless networks and analyze how wireless factors affect these metrics. We then present the research directions of deploying FL over wireless networks. Finally, open problems and industry interest of designing communication efficient FL are introduced.

A. Preliminaries of FL

Consider a set $\mathcal{U}$ of U devices orchestrated by a parameter server (PS) to jointly train a common ML model. We assume that each participating device i owns a dataset $\mathcal{K}_i$ of K_i training samples, where each training sample $k \in \mathcal{K}_i$ consists of an input vector $\mathbf{x}_{i,k}$ and a corresponding output vector $\mathbf{y}_{i,k}$. Next, we introduce different FL problems.

*1) **Common Federated Learning**:* The training objective of common FL is given as follows:

$$\min_{\mathbf{m}} \sum_{i=1}^{U} \frac{p_i}{K_i} \sum_{k \in \mathcal{K}_i} f\left(\mathbf{m}, \mathbf{x}_{i,k}, \mathbf{y}_{i,k}\right), \tag{1}$$

where $\mathbf{m} \in \mathcal{R}^d$ is the ML model that the devices aim to find collaboratively, $f(\cdot)$ is a loss function that captures the accuracy of the considered FL algorithm by building a relationship between an input vector $\mathbf{x}_{i,k}$ and the corresponding output vector $\mathbf{y}_{i,k}$; p_i is a scaling parameter that scales the weight of device i's average loss, $\frac{1}{K_i} \sum_{k \in \mathcal{K}_i} f\left(\mathbf{m}, \mathbf{x}_{i,k}, \mathbf{y}_{i,k}\right)$, on the total training loss with $\sum_{i=1}^{U} p_i = 1$. Problem (1) is commonly solved by using iterative distributed optimization techniques orchestrated by the PS. Federated Averaging (FedAvg) [34] is the first FL algorithm proposed by Google to solve problem (1). The training process of FedAvg at iteration t proceeds as follows:

a. The PS broadcasts the current global model $\mathbf{b}(t)$ to all (or a subset of) the devices.
b. Each device participating in this iteration uses some local learning method, such as the stochastic gradient descent (SGD), to train its ML model using locally available data (called local ML model).
c. Each device sends its updated ML model parameters, $\mathbf{m}_i(t+1)$, to the PS.
d. The PS updates global model as follows: $\mathbf{b}(t+1) = \sum_{i=1}^{U} p_i \left(\mathbf{m}_i(t+1) - \mathbf{b}(t)\right) + \mathbf{b}(t)$.
e. Steps from a. to d. are repeated for a certain number of iterations, or until some convergence criteria is met.

From the training procedure, we observe that, in FedAvg, each device transmits its model update $\mathbf{m}_i(t+1) - \mathbf{b}(t)$ to the PS instead of sending its private dataset, thus promoting data privacy for devices. Hereinafter, we define the implementation of steps from b. to d. as one *learning step*. Meanwhile, at step b., each device can update its ML model multiple times. Hereinafter, a device using the SGD method to update its ML model once is called one local update. Since FedAvg finds a common ML model for all the devices, the training loss of each device will be significantly increased when the data distribution of each device is non independent and identically distributed (Non-IID).

To deal with Non-IID data, next, we introduce personalized FL. In particular, we introduce two classical personalized FL algorithms: federated multi-task learning [35] and model agnostic meta learning (MAML) based FL [36].

*2) **Federated Multi-Task Learning**:* In federated multi-task learning (FMTL), devices are considered to implement correlated but different learning tasks. In other words, Non-IID data distributions of devices can be considered as different

tasks. The training purpose of FMTL is given as follows:

$$\min_{\mathbf{M},\mathbf{\Omega}} \sum_{i=1}^{U} \sum_{k\in\mathcal{K}_i} f\left(\mathbf{m}_i, \mathbf{x}_{i,k}, \mathbf{y}_{i,k}\right) + R\left(\mathbf{M}, \mathbf{\Omega}\right), \tag{2}$$

where $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_U]$, $\mathbf{\Omega}$ models the relationship among different learning tasks of devices, and function $R(\cdot)$ is a regularizer. To solve problem (2), one can use separate problem (2) into several subproblems so as to enable devices to solve problem (2) in a distributed manner. For example, the authors in [35] used a dual method and quadratic approximation to divide problem (2). Then, each device i can individually optimize its ML model $\mathbf{m}_i$ under given $\mathbf{\Omega}$ while the PS updates $\mathbf{\Omega}$ using the updated $\mathbf{M}$. After the devices and the PS iteratively optimize $\mathbf{M}$ and $\mathbf{\Omega}$, problem (2) can be solved.

From (1) and (2), we can see that, in FedAvg, devices will have the same ML model at convergence. In contrast, in FMTL, devices may have different ML models at convergence. This is due to the fact that for Non-IID data or different learning tasks, devices with different ML models can achieve less sum training loss than devices with a common ML model.

*3) **MAML Based FL**:* MAML based FL aims to find an ML model, using which each device can find a personalized ML model via one or a few steps of gradient descent iterations. The training purpose of MAML based FL is given as follows:

$$\min_{\mathbf{m}} \sum_{i=1}^{U} \frac{p_i}{K_i} \sum_{k\in\mathcal{K}_i} f\left(\mathbf{m} - \lambda \nabla f_i, \mathbf{x}_{i,k}, \mathbf{y}_{i,k}\right), \tag{3}$$

where λ is the learning rate and ∇f_i is the gradient descent of local ML model of device i. From (3), we can see that MAML based FL aims to find a common ML model for all devices. Then, the devices can use their own data to update their common ML models via a few steps of gradient descent so as to find their personalized ML models.

Given the overview of FedAvg, FMTL, and MAML based FL, we remark the following:

- FMTL directly optimizes the personalized ML model of each device while MAML based FL optimizes the initialization of ML model of each device.
- FedAvg is recommended for processing IID data while FMTL and MAML based FL are recommended for processing Non-IID data.
- Choosing between FMTL or MAML based FL depends on whether the PS knows the relationship among the data distributions of the devices.
- All FL algorithms must be trained by a distributed iterative process.

B. Performance Metrics of FL over Wireless Networks

Next, we introduce four key metrics that evaluate the performance of FL implemented over wireless networks: a) training loss, b) convergence time, c) energy consumption, and d) reliability.

*1) **Training Loss**:* Training loss is the value of the loss functions $f(\cdot)$ defined in (1), (2), and (3). From the FL training procedure, we can see that the FL training loss depends on the ML models of all devices. In wireless networks, devices' ML models are transmitted over imperfect wireless links. Therefore, they may experience transmission errors thus negatively impacting the training loss. Meanwhile, due to limited energy and computational capacity, only a subset of devices can participate in FL. Therefore, only a subset of devices' ML models can be used to generate the global ML model thus negatively impacting the training loss.

*2) **Convergence Time**:* For FL implemented over wireless networks, its convergence time T is expressed as

$$T = (T_{\mathrm{C}} + T_{\mathrm{T}}) \times N_{\mathrm{T}}, \tag{4}$$

where T_{C} is the time that each device used to update its local ML model at each learning step, T_{T} is the maximum ML model transmission time per learning step, N_{T} is the number of learning steps that FL needs to converge. From (4), we can see that FL convergence time depends on three components: a) ML parameter transmission delay T_{T}, b) the time T_{T} needed by each device to train its local ML model, and c) number of learning steps N_{T}. Here, we need to note that T_{C} and N_{T} are dependent. In particular, increasing the number of SGD steps to update a local ML model at each learning step (e.g., increasing T_{C}) can decrease the number of learning steps N_{T} that FL needs to converge.

*3) **Energy Consumption**:* The energy consumption E of each device participating the entire FL training is expressed as

$$E = (E_{\mathrm{C}} + E_{\mathrm{T}}) \times N_{\mathrm{T}}, \tag{5}$$

where E_{C} is energy consumption of each device training its ML model at each learning step and E_{T} is the energy consumption of transmitting ML parameters to the PS at each learning step. From (5), we can see that energy consumption of each device depends on three components: a) energy consumption for ML parameter transmission, b) energy consumption for training local ML model, and c) number of learning steps that FL needs to converge. Here, since increasing the number of SGD steps to update a local ML model at each learning step can decrease the number of learning steps N_{T} that FL needs to converge, a trade-off exists between E_{C} and N_{T}.

*4) **Reliability**:* FL reliability is defined as the probability of FL achieving a target training loss. For wireless FL, due to limited wireless resources, only a subset of devices can participate in the FL training at each learning step. Hence, the devices that transmit FL parameters to the PS at different learning steps may be different, which will affect the FL convergence time and training loss. Meanwhile, imperfect wireless links will cause errors on the FL parameters used to generate the global ML model, hence decreasing training loss.

C. Effects of Wireless Factors on FL Metrics

Given the metrics defined in the previous subsection, we first explain how wireless network factors such as spectrum, transmit power, and computational capacity affect these FL metrics. Table I summarizes the relationship between various wireless factors and FL performance metrics. In Table I, a tick implies that the communication factor will affect the FL

TABLE I
SUMMARY OF EFFECTS OF COMMUNICATION FACTORS ON FL METRICS

Communication Factors	Training Loss	Local Training Time T_{C}	FL Parameter Transmission Time T_{T}	Total Number of Learning Steps N_{T}	Energy Used for Local Training E_{C}	Energy Used for FL Transmission E_{T}	Reliability
Spectrum resource	√		√	√		√	√
Computational capacity	√	√		√	√		
Transmit power	√		√	√		√	√
Wireless channel	√		√	√		√	√
Set of devices that participate in FL	√		√	√			√
Size of the FL parameters trained by each device	√	√		√	√		√
Size of the FL parameters transmitted by each device			√			√	

performance metrics. For example, the spectrum resource allocated to each device for FL parameter transmission will affect the training loss, FL parameter transmission time per learning step T_{C}, Energy consumption of FL parameter transmission E_{C}, and reliability of FL. Next, we explain how these wireless factors affect the FL performance metrics as follows:

- Spectrum resource allocated to each device determines the signal-to-interference-plus-noise ratio (SINR), data rate, and the probability that the transmitted FL parameters include errors. Hence, spectrum resource affects the training loss, T_{T}, E_{T}, and reliability.
- Computational capacity determines the number of SGD updates that each device can perform at each learning step. Hence, computational capacity affects the time and energy used for local training. Meanwhile, as the number of SGD updates decreases, the training loss increases and the number of learning steps that FL needs to converge increases.
- Transmit power and wireless channel determine the SINR, data rate, and the probability that the transmitted FL parameters include errors. Therefore, as the transmit power of each device increases, the training loss, T_{T}, N_{T}, and reliability decrease but E_{T} increases.
- In FL, as the number of devices that participate in FL increases, the training loss and N_{T} decrease while T_{T} and reliability increase.
- As the size of the FL parameters trained by each device increases, the FL training loss, reliability, and the total number of learning steps may decrease. However, the energy and time used for training FL model increases.

D. Research Directions of Deploying FL over Wireless Networks

Next, we present a comprehensive overview on the key research directions that must be pursued for practically deploying FL over wireless networks. For each research direction, we first outline the key challenges, and then we discuss the state of the art, while also providing a recent result.

1) ***Compression and Sparsification****:* A major challenge in distributed learning, particularly over wireless channels, is the communication bottleneck due to the large size of the trained models. For emerging DNNs with hundreds of millions of training parameters, transmitting so many locally trained parameter values from each participating device to the PS at every iteration of the learning algorithm over a shared wireless channel is a significant challenge.

We would like to note here that the transmission of locally trained model parameters to the PS over a noisy wireless channel is a joint source-channel coding problem. Indeed, considering the fact that the PS is interested in the average of the models, rather than the individual model updates from different devices, this can be classified as a joint source-channel function computation problem [37], [38]. In general, we do not have an optimal solution to such a problem, particularly in the practical finite blocklength regime. The conventional approach to this problem is to separate the compression of DNN parameters from the transmission over the channel. This so-called 'digital' approach converts all the local updates into bits, which are then transmitted over the channel as reliably as possible, and all the decoded 'lossy' reconstructions are averaged by the PS. A more efficient method would be to directly map each locally trained model parameter to a channel input in an 'analog' fashion [13]. While we will explore this approach in Section III in detail, here we focus on digital schemes, and assume that each device individually compresses its own parameters.

Numerous communication efficient learning strategies have been proposed in the ML literature to reduce the amount of information; that is, the number of bits exchanged between the devices and the PS per global iteration. We classify these approaches into two main groups; namely *sparsification* and *quantization*. We would like to highlight that, thanks to the separation between compression and transmission of compressed bits to the PS, these strategies are independent of the communication medium and the communication protocol employed to exchange model updates between the devices and the PS, as they mainly focus on reducing the size of the messages exchanged. Therefore, these techniques can be incorporated into the resource allocation and device selection policies that will be presented below.

The objective of sparsification is to transform the d-dimensional model update $\mathbf{m}$ at a device to its sparse representation $\tilde{\mathbf{m}}$ by setting some of its elements to zero. Sparsification can also be considered as applying a d-dimensional mask vector $\mathbf{M} \in \{0,1\}^d$ on $\mathbf{m}$, such that $\tilde{\mathbf{m}} = \mathbf{M} \otimes \mathbf{m}$, where $\otimes$ denotes element-wise multiplication. We can define the *sparsification level* of this mask by $\phi \triangleq ||\mathbf{M}||_1/d$, i.e., the ratio of its non-zero elements to its dimension. Note that, when conveying a sparse model update to the PS, rather than conveying the

values of all d values of the model update, each device needs to convey only the values of ϕd non-zero values and their locations. Therefore, the lower the sparsification level, the higher the compression ratio, and the lower the communication load. It is known that when training a complex DNN model using stochastic gradient descent methods, model updates can be highly sparse. Indeed, it has been shown that when training some of the popular large-scale architectures, such as ResNet [39] or VGG [40], sparsification levels of $\phi \in [0.01, 0.001]$ provides significant reduction in the communication load with almost no loss in their generalization performance [41], [42].

Top-K sparsification is probably the most common strategy used in distributed learning. In top-K sparsification, each device constructs its own sparsification mask $\mathbf{M}_{i,t}$ at each iteration by identifying the K values in its local update with the largest absolute values [27]–[29]. A simpler alternative to top-K is rand-K sparsification [29], which selects the sparsification mask $\mathbf{M}_{i,t}$ randomly from the set of masks with sparsification level K. Both rand-K and top-K are biased compression strategies. In the case of rand-K, unbiased model updates can be obtained by scaling $\mathbf{M}_{i,t}$ with d/K, albeit at the expense of increasing the variance, which is not desirable in practice [29]. Top-K sparsification has been shown to outperform rand-K in practical applications in terms of both the test accuracy and the convergence speed; however, top-K sparsification requires sorting the elements of the model update vector at each iteration, which can significantly slow down the learning process. Moreover, as mentioned above, top-K sparsification requires transmitting the location of the non-zero values within the model update vector, which increases its communication load, whereas this is not needed for rand-K if a pseudo-random generator with a common seed is used across all the devices to generate the same mask. A time-correlated sparsification strategy is introduced in [43], where a common mask is sent from the PS at each iteration to be employed by all the devices to remove the additional communication load due to sending locations of the non-zero values, and instead, each device sends only a limited number of significant values that are not present in this common mask, enabling exploration of more efficient masks. This approach exploits the time correlations between model updates across different iterations, and can provide up to 2000 times reduction in the communication load with minimal loss in model accuracy. We also note that, when employed for distributed training of DNN architectures, these sparse communication strategies can be applied to each layer of the network separately, since it is observed that different layers have different tolerance to sparsification of their weights [41], [43].

As mentioned above, the weights of a DNN take values from real numbers, and hence, even after sparsification they cannot be transmitted to the PS as they are, and must be quantized. In practice, since even the computing of local iterations are carried out using 32bit floating point representations, we can assume that each weight can be conveyed to the PS perfectly using 32 bits. Quantization techniques aim at identifying more efficient representations of the network weights that use less than 32 bits per weight [24]–[26]. At the extreme, only a single bit can be used to represent only the sign of each element, which would result in a 32 times reduction in the communication load. Sign based compression techniques for distributed optimization have been studied for a long time mainly to improve the robustness and convergence of learning algorithms [44]. It has been recently shown that simple sign-based quantization together with majority voting converges to the optimal solution (under certain assumptions), and provides an extremely communication-efficient viable alternative in practice as well [45]–[47]. A more advanced vector quantization scheme is considered in [48].

State of the Art: Next, we discuss a number of recent works on the use of compression and sparsification techniques for deploying FL over wireless networks. The authors in [49] studied the use of compression and sparsification techniques for local ML model transmission and analyzed their convergence properties in both homogeneous and heterogeneous local data distributions settings. In [50], the authors studied the use of lossy compression for global ML model parameter transmission. The work in [51] introduced a ternary quantization approach for the training and inference stages of devices. The authors in [52] investigated the fundamental trade-off between the number of bits needed to encode compressed vectors and the compression error and performed both worst-case and average-case analysis on the FL convergence. In [53], the authors designed a hyper-sphere quantization based FL algorithm so as to achieve a continuum of trade-offs between communication efficiency and gradient accuracy. The authors in [54] focused on the design and analysis of physical layer quantization and transmission methods for wireless FL and evaluated the impact of various quantization and transmission options of the ML models on the learning performance. The work in [55] designed a novel FL algorithm based on random linear coding and developed efficient power management and channel usage techniques to manage the trade-offs between power consumption, communication bit-rate and convergence rate. In [56], the count sketch is used to compress the local ML parameters thus overcoming the challenges of sparse device participation while still achieving high compression rates and convergence speed. Additional forms of probabilistic scalar quantization for FL were considered in [57]–[60].

We would like to highlight that, most of the literature on distributed learning, and particularly its implementation over a wireless network, focus on the limitation of the uplink resources, and study quantization and sparsification of model updates from the devices while assuming that the global model from the PS is conveyed perfectly to all the participating devices. However, in the case of bandwidth-limited wireless networks, broadcasting the global model to all the wireless devices can be a challenge as well. The convergence of FL with noisy downlink transmission of the global model is studied in [61], and both digital and analog transmission of global model updates is considered.

Representative Result: Next, we show a representative work in [48] that investigates the use of universal vector quantization for FedAvg. In particular, in their considered model, each device transmits its gradient parameters over a flat fading channel with Gaussian noise and interference. To speed up convergence, the authors first design a probabilistic

device selection scheme based on the ML gradient parameters as well as the distances between the PS and the devices. Given the probabilistic device selection scheme, the PS can select a subset of devices for local FL parameter transmission and optimize resource allocation for the selected devices. To further reduce the FL convergence time, the authors study the use of universal vector quantization to compress the local FL model parameters of the selected devices, and particularly, to represent their FL parameters using a limited number of bits, such that the PS can still accurately generate the global ML model via FL parameter aggregation. The motivation for using universal vector quantization for FedAvg is given as follows. First, due to the heterogeneous nature of the training data available at the devices, *a-priori* knowledge of the underlying distribution of the local FL model parameters is not available at the device side, which motivates compressing the local FL model as a form of universal quantization. Second, the fact that the PS and the devices communicate repeatedly allows them to share a source of common randomness, e.g., a random seed, enabling the participating devices to implement low-distortion discretization in a universal manner via random lattice quantization.

The resulting FL parameter compression scheme consists of the following steps: As a preliminary step, an L-dimensional lattice is fixed. Upon FL model parameter transmission, each device normalizes its local FL model parameters. The normalized result is divided into several L-sized vectors, to which the devices add a random dither signal randomized in an i.i.d. fashion from a uniform distribution over the basic cell of the lattice. The dithered signal is discretized by projecting to the nearest lattice point, and the discrete quantity is further compressed prior to transmission using lossless entropy coding. The PS decompresses the model updates by recovering the lattice point, and subtracting the dither signal from it. The fact that the devices and the PS can generate the same dither signals relies upon their ability to share a source of common randomness. The analysis of convergence of the universal vector quantization based FL is shown in Theorem 1 of [48], which shows that the quantization error is mitigated by averaging the local FL parameters at the PS. This is another reason that the authors used the universal vector quantization for FL parameter compression.

Fig. 2 shows the convergence of the FL algorithms that use various quantization methods. The universal vector quantization scheme with two-dimensional lattices is compared to its implementation with conventional scalar quantizers, the QSGD FL algorithm proposed in [62], and the combination of uniform quantization with random rotations proposed in [63]. From Fig. 2, we see that the universal vector quantization based FL outperforms the other quantization based FL methods. This is because the universal vector quantization based FL implements substractive dithered lattice quantization, which reduces the distortion induced by quantizing the FL model parameters and mitigates its effect on the aggreagated global model.

*2) **Wireless Resource Management**:* As shown in Table I, wireless resources such as spectrum, transmit power, and computational capabilities jointly determine the FL training loss,

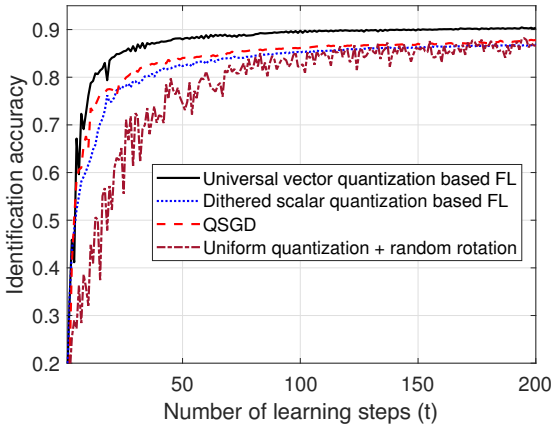


Fig. 2. Comparison of various quantization based FL [48].

convergence time, energy consumption, and reliability. Due to limited resources in wireless networks, it is necessary to optimize resource allocation so as to enable wireless networks to efficiently complete the FL training process. However, analyzing the effects of resource allocation on the FL performance faces several challenges. First, FL training process is distributed and iterative, but it is challenging to quantify how each single model update affects the entire training process. Also, since each device only exchanges its gradient vector with the PS, the PS does not have any information about devices' local datasets, and cannot use sample distribution or the values of the data samples to decide how resource allocation will affect the FL convergence.

State of the Art: Now, we discuss a number of recent works on the optimization of spectrum resources for deploying FL over wireless networks. In [12], the authors considered the implementation of FL over a hierarchical network architecture and they showed that the global convergence can be accelerated if local training is enabled with the help of small base stations (BSs), which only occasionally communicate with the macro BS for global consensus. Meanwhile, local learning not only speeds up the learning process, but also reduces the energy consumption of communication due to short distance transmissions, and increases communication efficiency by frequency reuse across multiple small cells, enabling parallel local learning processes. The authors in [64], [65] study the trade-off between the local ML model updates and global ML model aggregation so as to minimize the total energy consumption for local ML model training and transmission or the FL training loss. The authors in [66] study the use of gradient statistics to optimize the set of devices that participate in FL at each training round. The authors in [67] assume that the local FL model transmitted by the device can be decoded by the PS only when the SINR is under the target threshold, and analyzed how user scheduling affects the FL convergence. The work in [68] designed a FL algorithm which can handle heterogeneous user data without further assumptions except strongly convex and smooth loss functions and then optimized the resource allocation to improve FL convergence and training loss. The authors in [69] jointly optimized device scheduling and resource allocation policies to

maximize the model accuracy within a given total training time budget for latency constrained wireless FL. In [70], the authors designed a multi-armed bandit based algorithm to select the devices that must participate in FL without knowing wireless channel state information and statistical characteristics of devices. In [71], data-driven experiments are designed to show that different temporal device selection patterns lead to considerably different learning performance. With the obtained insights, device selection and bandwidth allocation are jointly optimized utilizing only currently available wireless channel information.

Representative Result: In [11], the authors analyzed how spectrum resource allocation, user selection, and transmit power of each device affects the FL convergence and optimized these wireless factors to improve FL training loss. In particular, the authors considered the implementation of an FedAvg algorithm over a wireless network that consists of multiple edge devices and one BS. At each learning step, edge devices will train the local ML models and send the ML parameters to the BS. The BS acts as a PS to aggregate the received ML parameters so as to generate the global ML model and send the model back to all devices. In the considered model, all ML parameters are transmitted over wireless flat fading channels with Gaussian noise and interference. Therefore, the imperfect wireless transmission will cause errors on the local ML models received by the BS. To analyze how imperfect wireless transmission affects the FL convergence, the authors built the relationship between local ML model transmission and SINR of each wireless link. First, the authors assumed that the local ML parameters of each device is transmitted as a single packet. Hence, the imperfect wireless transmission may cause errors on the transmitted packets. Meanwhile, the authors assumed that the BS will directly abandon the erroneous local ML parameters and will not use them for the global ML model generation. To this end, the authors can use SINR to derive the probability (called packet error rate) of each local ML packet including errors caused by wireless transmission. Given this probability, one can analyze the expected FL convergence, as follows:

$$\mathbb{E}\left(F\left(\mathbf{b}_{t+1}\right)-F\left(\mathbf{b}^{*}\right)\right) \leq A^{t}\mathbb{E}\left(F\left(\mathbf{b}_{0}\right)-F\left(\mathbf{b}^{*}\right)\right) + \underbrace{\frac{2\zeta_{1}}{LK}\sum_{i=1}^{U}K_{i}\left(1-a_{i}+a_{i}q_{i}\left(\mathbf{r}_{i},P_{i}\right)\right)\frac{1-A^{t}}{1-A}}_{\text{Impact of wireless factors on FL convergence}}, \quad (6)$$

where $A < 1$ is the convergence rate function of network parameters such as power, bandwidth, and link quality (see [11, Theorem 1] for expression), $F\left(\mathbf{b}\right) = \sum_{i=1}^{U}\frac{p_i}{K_i}\sum_{k\in\mathcal{K}_i} f\left(\mathbf{b},\mathbf{x}_{i,k},\mathbf{y}_{i,k}\right)$, $a_i, \mathbf{r}_i, P_i$ are respectively the device selection index, resource allocation vector, and transmit power of device i, $q_i\left(\mathbf{r}_i,P_i\right)$ is the probability of the transmitted packet of device i including errors, $\mathbf{b}_{t+1}$ is the global ML model at learning step $t+1$ while $\mathbf{b}^*$ is the optimal global ML model that can solve problem (1). From (6), we can see that wireless factors (e.g., $a_i, \mathbf{r}_i, P_i$) and FL parameters (e.g., K_i, U,) jointly determine the FL convergence. From (6), we can also see that as t is large enough, $A^t\mathbb{E}\left(F\left(\mathbf{b}_0\right)-F\left(\mathbf{b}^*\right)\right)=0$ but the second term will not be equal to 0. Therefore, we only need to minimize the second term via optimizing resource allocation and device selection.

Equation (6) captures how the packet error rates and device selection affect the FL convergence. In a special scenario where all devices can participate in FL and all transmitted ML parameters are correct, the FL algorithm can find an optimal global ML model to solve problem (1). According to this equation, one can analyze the effects of other wireless factors (e.g., device mobility, energy harvesting) that is related to packet error rates on the FL convergence.

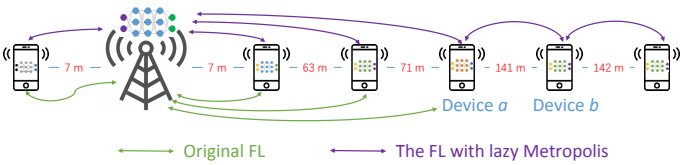
*3) **FL Training Method Design**:* Beyond the use of wireless techniques, one can design novel FL training methods and adjust the learning parameters (e.g., step size) to enable FL to be efficiently implemented over wireless networks. Naturally, wireless devices have a limited amount of energy and computational resources for ML model training and transmission. In consequence, the size of ML model parameters that can be trained and transmitted by a wireless device is typically small and the time duration that the wireless devices can be used for training FL is typically short. Hence, while designing FL training methods, the energy, computation, and training time constraints need to be explicitly taken into account. Meanwhile, FL training methods determine the network topologies formed by the devices thus significantly affecting the FL training complexity and the FL convergence time. In consequence, designing FL training methods also needs to jointly consider the locations and mobility patterns of wireless devices as well as wireless channel conditions.

State of the Art: Designing communication efficient FL training methods has been studied from various perspectives. In particular, an error feedback based SignSGD update method is proposed in [47] to improve both convergence and generalization. In [12], hierarchical FL is proposed, where devices are grouped into clusters, and devices within each cluster carry out local learning with the help of a small BS or a cluster head, while a global model is trained at the macro BS. This framework is extended in [72], which designed a training method for a multi-layer FL network. The authors in [73] and [74] proposed a gradient aggregation method so as to decrease the number of devices that must transmit the local ML parameters to the PS thus reducing the FL communication overhead. In [75], the authors introduced a non-parametric generalized Bayesian inference framework for FL so as to reduce the number of learning steps that FL needs to converge. The authors in [76] proposed a post-local SGD update method that enables each device to update its ML parameters once in the initial multiple learning steps while update its ML parameters several times in the following learning steps. This post-local SGD method can significantly improve the generalization performance and communication efficiency. The work in [77] designed a parallel restarted SGD method using which each device will average its ML model every certain learning steps and perform local SGDs to update its ML model in other learning steps. In [78], the authors designed a personalized FL algorithm using Moreau envelopes as each device's regularized loss function which can decouple personalized ML model optimization from the global model

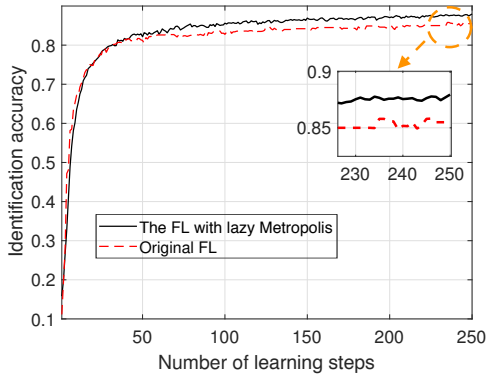
learning in a bi-level problem stylized for personalized FL. The work in [79] addressed the FL problem, in which the users are distributed and partitioned into clusters. In particular, the authors proposed a new framework dubbed the iterative federated clustering algorithm, which alternately estimates the cluster identities of the users and optimizes model parameters for the user clusters via gradient descent. In [80], the authors studied FL over wireless device-to-device networks by providing theoretical insights into the performance of digital and analog implementations of decentralized stochastic gradient descent. The authors in [81] designed a novel FL optimization objective inspired by fair resource allocation in wireless networks that encourages more uniform accuracy distributions across devices. The work in [82] developed a one-shot unsupervised federated clustering scheme based on the Lloyd's method used for k-means clustering.

Representative Result: To include more devices to participate in FL and reduce the devices' reliance on the PS, the authors in [83] used decentralized averaging methods to update the local ML model of each device. In particular, using the decentralized averaging methods, each device only needs to transmit its local ML parameters to its neighboring devices. Each device can use the ML parameters of its neighboring devices to estimate the global ML model. Therefore, using the decentralized averaging methods can reduce the communication overhead of FL parameter transmission. Meanwhile, since each device only needs to connect to its neighboring devices, the devices that cannot connect to the PS due to limited energy and wireless resources may be able to be associated with their neighboring devices so as to participate in FL training. Therefore, using the decentralized averaging methods for local ML model update can include more wireless devices to participate in FL training. Meanwhile, using decentralized averaging methods, the devices can form different network topologies to further improve FL parameter transmission time and ML model inaccuracy caused by imperfect wireless transmission. Finally, since each device shares its ML parameters with only its neighboring devices and the PS cannot know the ML parameters of all devices, privacy against the PS can be improved (assuming the neighbouring devices are trustworthy).

To show the performance of the FL with the decentralized averaging method, particularly the lazy Metropolis update method (called the FL with lazy Metropolis hereinafter), we implemented a preliminary simulation for a network that consists of one BS that is acted as a PS and six devices, as shown in Fig. 3(a). In Fig. 3(a), the green and purple lines respectively represent the local ML parameter transmission of original FL and the FL with lazy Metropolis. Due to the transmission delay requirement, only 4 devices can participate in original FL. For the FL with the lazy Metropolis update method, 6 devices can participate in the FL training process since the devices that use the lazy Metropolis update method can connect to their neighboring devices. Therefore, from Fig. 3(b), we can see that the FL with lazy Metropolis outperforms the original FL in terms of identification accuracy. In fact, the FL with lazy Metropolis can also reduce the energy consumption for device b since it only needs to transmit its ML model parameters to device a instead of the BS.



(a) Simulation system



(b) Simulation result

Fig. 3. Simulation system and result to show the performance of the FL with the lazy Metropolis update method. In this figure, a red digit is the distance between two adjacent devices [83].

E. Open Problems of Deploying FL over Wireless Networks

Given the general research directions and challenges of deploying FL over wireless networks, next, we discuss open research problems.

*1) **Convergence Analysis**:* FL convergence analysis results show the effects of wireless factors on key learning metrics; and hence, can be used to optimize the allocation of wireless resources and on deciding other wireless system parameters. For convergence analysis, there is a need to analyze how wireless factors affect the convergence of realistic FL with non-convex local ML models and loss function. Most existing works [11], [64]–[67] use distributed optimization methods to analyze the effects of wireless factors on the FL convergence, assuming that the FL loss function is strongly convex and twice-continuously differentiable, and its gradient is uniformly Lipschitz continuous. However, realistic FL algorithms may not satisfy these conditions. Meanwhile, the convergence analysis should characterize the dynamics caused by SGD updates for local ML models, wireless channels, and device mobility. In addition, instead of finding the upper and lower convergence bounds in the existing works, the designed convergence analysis methods must find an exact convergence value and show the exact number of learning steps needed to converge.

*2) **Wireless Resource Management**:* While there have been an increasing number of studies on the optimization of wireless resource allocation for FL, there are still several many open problems, including: 1) considering the optimization of resource allocation based on the mobility patterns of devices, 2) jointly considering resource allocation, compression scheme design, and learning parameter (e.g. step size) adjustment so as to simultaneously reduce the time used for ML parameter

training and transmission, 3) optimizing resource allocation for the devices that participate in FL, while guaranteeing the quality-of-service of other cellular-connected devices, and 4) adopting suitable frequency bands (e.g., mmWave and Thz bands) for local and global ML parameter transmission.

*3) **Compression and Sparsification**:* For developing compression and sparsification schemes to improve FL performance metrics, there are several key problems. First, in wireless networks, link characteristics of each device will be different (e.g., different data rates). Hence, to efficiently use wireless resources for FL model transmission, it is necessary to design novel heterogeneous compression schemes that enable each device to encode its local FL model using different number of bits or different coding techniques. Second, since one can use gradient vectors to recover the raw data, called *gradient leakage* [84], it is necessary to design new compression or sparsification schemes that optimize FL performance metrics while considering data leakage. Although designing complicated compression or sparsification schemes can significantly reduce data leakage, it also introduces processing latency. Therefore, there is a need to design new compression or sparsification schemes that can significantly reduce data leakage while reducing FL convergence time.

*4) **FL Training Method Design**:* Designing efficient FL training methods requires addressing a number of key problems. A fundamental problem is to enable the devices to form an optimal network topology that maximizes various FL performance metrics and trade-offs. This is a challenging problem since the solution must jointly account for the network topology, device heterogeneity, wireless dynamics, FL learning parameters (e.g., the data size of local ML model), and multiple dependent FL performance metrics. Other important open problems include: 1) designing asynchronous training methods while considering the network topology optimization, 2) designing FL training methods for devices that may not completely know the network architecture, other device locations, and network composition; and hence, can connect only to a limited number of devices, 3) designing mobility-aware FL training methods, and 4) designing FL training methods that optimize FL performance metrics over wireless links while preventing data leakage.

F. Industry Interest

As we have already mentioned, centralized based algorithms cannot fulfil the low latency demands of near real time applications of 5G and beyond cellular networks, while at the same time satisfying security and privacy requirements. Therefore, approaches that keep local data on resource-constrained edge nodes (such as mobile phones, IoT devices or radio sites) and employ edge computation to learn a shared model for prediction have become increasingly attractive for the networking and IoT industry, and in recent years it have appeared several implementations of distributed ML.

In April 2017, Google published a blog post [85] describing they had successfully tested an FL method with many Android mobile devices. Using a federated averaging algorithm, a global model had been trained and deployed on Android mobiles to suggest search queries based on typing context from Android Gboard. The mobile used the model stored on the device to predict search queries (such as suggesting next words and expressions) but training and model update would only take place once the mobile was connected to WiFi and charging. As such it was ensured that only the user has a copy of their data.

Besides Google, many other industrial researchers have also recently started exploring FL. Intel [86] used FL to do medical imagining where personal data used for training a global model is kept local. During MWC 2019 ByteLake and Lenovo [87] have demonstrated FL IoT industry application that enables IoT devices in 5G networks to learn from each other as well as makes it possible to leverage local ML models on IoT devices.

As we discuss in this paper, despite the apparent opportunities FL offers in wireless networks it is still in its early stages, as there exist several critical challenges that need to be researched, especially for large scale telecom application, such as computational resource allocation for training FL models at edge devices, selection of users for FL, energy efficiency of FL implementation, spectrum resource allocation for FL parameter transmission, and design of communication-efficient FL. Nevertheless, the telecom industry has recently started industrially applying distributed ML to improve privacy when using ML for network optimization, time-series forecasting [88], predictive maintenance and quality of experience (QoE) modeling [89], [90]. To better understand the potential of FL in a telecom environment, the Ericsson authors in [90] have tested it on a number of use cases, migrating the models from conventional, centralized ML to FL, using the accuracy of the original model as a baseline. Their research has indicated that the usage of a simple neural network results in a significant reduction in network utilization, due to the sharp drop in the amount of data that needs to be shared. Besides being improved by 5G techniques, FL has also been integrated in the 5G Network Data Analytics (NWDA) architecture where it has been used to deal with 5G problems such as Network Data Analytics Function (NWDAF) [91], in order to improve privacy.

III. FL with Over-the-Air Computation

As discussed earlier, one challenge confronting the implementation of FL in wireless networks, called federated edge learning (FEEL), is to overcome the communication bottleneck, which arises from many devices uploading high-dimensional model updates (locally trained models or stochastic gradients) to a PS. Researchers have attempted to reduce the resultant communication latency using different approaches such as excluding slow devices (“stragglers") [92], [93], selecting only those devices whose updates can significantly accelerate learning [73], [94], or compressing updates by exploiting their sparsity using the techniques outlined in Subsection II-D1. An alternative approach of our interest in this section is to design new multiple access schemes targeting FEEL. The main drawback of the classic orthogonal-access schemes (e.g., OFDMA or TDMA) is that they do not scale well with the number of devices. Specifically, the

required radio resources increase linearly with the number of transmitters, or else the latency will grow linearly. A recently emerged approach, called over-the-air computation, which is also known as AirComp, can provide the needed scalability for multi-access in FEEL. Fundamental limits of OAC have been studied in [95], [96] from an information theoretic perspective. A similar idea was considered in [97] for the distributed estimation of a discrete variable. OAC is applied to wireless communications in [98]–[100]. Specifically, the deployment of AirComp to support FEEL, termed AirComp-FEEL, exploits the wave-form superposition property of a multi-access channel together with simultaneous transmission to realize over-the-air model/gradient aggregation [13]–[15]. Given simultaneous access, the latency becomes independent of the number of devices. This overcomes the communication bottleneck to facilitate the implementation of FEEL over many devices. In this section, we shall first discuss the basic principle and techniques of AirComp, and then explore its deployment in a communication-efficient FEEL system.

A. AirComp Principle and Techniques

*1) **AirComp Principle**:* The mentioned idea of AirComp is elaborated as follows. Given simultaneous time-synchronized transmission by devices, their signals are superimposed over-the-air and their weighted sum, called the aggregated signal, is received by the PS, where the weights correspond to the channel coefficients. For AirComp-FEEL, it is desirable to have uniform weights so that the aggregated signal is not biased towards any device, and can be easily converted to the desired average of the transmitted signals (i.e., model updates). To make this possible requires each device to modulate its signal using linear analog modulation and to invert its fading channel by transmission-power control. The former operation is necessary to exploit the channel's analog-waveform superposition property and the latter aligns the received magnitudes of individual signal components, called magnitude alignment. One may question the optimality of the use of seemingly primitive analog modulation compared with sophisticated digital modulation and coding. Interestingly, from the information-theoretic perspective, it was shown in [37] that AirComp can be optimal in terms of minimizing the mean squared error (MSE) distortion if all the multi-access channels and sources are Gaussian and independent.

Though FEEL requires only over-the-air averaging, AirComp is capable of computing a broad class of so called nomographic functions [38], [98]. They are characterized by a post-processing function of a summation form with each term being a pre-processing function of an individual data sample. Besides averaging, other examples include arithmetic mean, weighted sum, geometric mean, polynomial, and Euclidean norm. Consequently, except for averaging, the implementation of AirComp of a nomographic function usually requires pre-processing of data before transmission and post-processing at the receiver. For a general function, it can be decomposed as a summation form of nomographic functions [101]. This suggests the possibility of approximately computing a general function with AirComp.

A key requirement for implementing AirComp is time synchronization of devices' transmissions. Such requirements also exist for uplink transmission (e.g., TDMA and SC-FDMA) in practical systems (e.g., LTE and 5G). In such systems, a key synchronization mechanism is called "timing advance", which can be also adopted for AirComp synchronization. The technique of timing advance involves each device estimating the corresponding propagation delay and then transmitting in advance to "cancel" the delay. Thereby, different signals can arrive at the BS in their assigned slots (in the case of TDMA) or overlap with sufficiently small misalignment (in the cases of SC-FDMA and AirComp). Considering a synchronization channel for the purpose of propagation-delay estimation, its accuracy is proportional to the channel bandwidth [102]. For instance, the estimation error is no larger than 0.1 microsecond for a bandwidth of 1 MHz. If AirComp is deployed in a broadband OFDM system (see the next sub-section), the error gives rise to only a phase shift to a symbol received over a sub-channel so long as the error is shorter than the cyclic prefix (CP). Then the phase shift can be compensated by sub-channel equalization. In an LTE system, the CP length is several microseconds, and hence more than sufficient for coping with synchronization errors. This suggests the feasibility of AirComp deployment in practical systems. The impact of potential remaining synchronization errors on the performance of AirComp and techniques to tackle them have been recently studied in [103].

The distortion of digital modulation originates from quantization and decoding errors. In contrast, for AirComp, the main source of signal distortion is channel noise and interference that directly perturb analog modulated signals. Hence, a commonly used performance metric for AirComp is the MSE distortion of received functional values with respect to the ground-truth. In the context of AirComp-FEEL, channel noise and interference perturb the model updates and their effects can be evaluated using the relevant metric of learning performance. Finally, it is worth mentioning that AirComp is similar to non-orthogonal multiple access (NOMA) in both being simultaneous-access schemes. However, the distinction of AirComp is the harnessing of "inference" for functional computation via devices' cooperation. On the other hand, NOMA attempts to suppress inference as the devices (subscribers) transmitting independent data compete for the use of radio resources.

*2) **Broadband AirComp**:* In a practical broadband system, the spectrum is divided into sub-carriers using the OFDM modulation. The deployment of AirComp-FEEL in such a system involves simultaneous over-the-air aggregation of model-update coefficients transmitted over sub-carriers subject to power constraints of individual devices. The channel inversion discussed in the preceding sub-section need be generalized to the case of multiple sub-carriers as follows. Consider a specific uploading device. For each OFDM symbol, ideally each sub-carrier is linearly analog modulated with a single model/gradient element, whose power is determined by channel inversion. However, due to the power constraint, it is impractical to invert those sub-carriers in deep fade; hence they are excluded from transmission, called *channel truncation*.

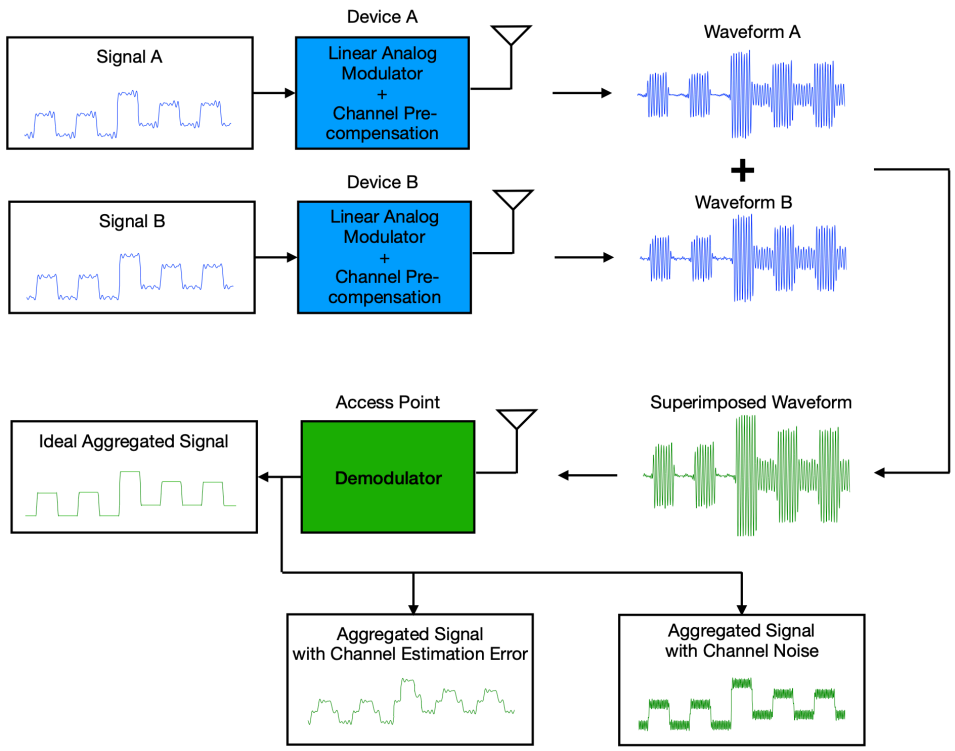


Fig. 4. The principle of AirComp [100].

AirComp requires all devices to have fixed and identical mappings between update coefficients to sub-carriers. As a result, channel truncation results in the erasure of coefficients mapped to sub-carriers in deep fade as they cannot be remapped to other sub-carriers. Channel truncation can potentially have a near-far problem where the fraction of erased coefficients, called truncation ratio, is much larger for a nearer device from the PS (hence with larger severe path loss) than a faraway device (will smaller loss). The problem introduces bias and degrades the learning performance. One solution is to apply channel truncation based only on small-scale fading with two-fold advantages: 1) approximately equalizing truncation ratios among devices, and 2) allowing the PS to exploit data even at faraway devices. The resultant scheme of truncated channel inversion scales the symbol transmitted over the m-th sub-carrier by a coefficient $p_k^{(m)}$ given as:

$$p_k^{(m)} = \begin{cases} \dfrac{\eta}{r_k^{-\frac{\alpha}{2}} h_k^{(m)}}, & |h_k^{(m)}|^2 \geq g_{\text{th}} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $r_k^{-\frac{\alpha}{2}}$ is the path loss and $h_k^{(m)}$ the fading gain. The parameter η represents the aligned received magnitude of different signal components and is chosen by observing individual power constraints of all devices. Next, given truncated channel inversion, the PS demodulates a certain number of OFDM symbols and thereby receives from the sub-carriers an over-the-air aggregated model update. This is then used to update the global model.

*3) **MIMO AirComp**:* MIMO (or multi-antenna) communication is widely adopted in practical systems (e.g., LTE and 5G) to support high-rate access by spatial multiplexing of data streams. The deployment of AirComp-FEEL in a MIMO system can leverage spatial multiplexing to reduce the communication latency by a factor equal to the multiplexing gain. Realizing the benefit requires the design MIMO AirComp, a technique multiplexing parallel over-the-air aggregation or equivalently AirComp of vector symbols, each comprising multiple update coefficients. The main distinction of MIMO AirComp is the use of receive beamforming, called aggregation beamforming, to enhance the received signal-to-noise ratios (SNRs) of aggregated observations from the PS array. The intuition behind the design of aggregation beamforming is that in terms of subspace distance, the beamformer should be steered aways from the relatively strong MIMO link and closer to those relatively weaker links. The purpose is to enhance the received SNRs of the latter at the cost of those of the former, thereby equalizating their channel gains. This facilitates the subsequent spatial magnitude alignment to enhance post-aggregation SNRs. Given the aggregation beamformer, the effective MIMO channel can be inverted at each device to implement spatial magnitude alignment after aggregation beamforming. Finding the optimal aggregation beamformer is a non-convex program and intractable. An approximate solution can be found in closed form though to mathematically express the above design intuition. Specifically, the received SNRs of spatial data streams from an individual MIMO link as observed after aggregation beamforming can be approximated using the smallest SNR, corresponding to the weakest eigenmode of the effective channel. Using the approximation, an approximate of the optimal aggregation beamformer can be

obtained as the first L left eigenvectors of the following matrix: $\mathbf{G} = \lambda_{\min,k}^2 \mathbf{U}_k \mathbf{U}_k^H$ where $\lambda_{\min,k}^2$ is the smallest singular value of the k-th link and $\mathbf{U}_k$ its left $L \times 1$ eigen subspace [38]. The matrix suggests that the aggregation beamformer is a weighted centroid of the eigen subspaces of individual MIMO links, where the weights are their smallest eigenvalues. This is aligned with the design intuition mentioned above.

B. Design of AirComp Federated Learning

Consider the AirComp-FEEL system in Fig. 5. In this section, we discuss several issues concerning the design of such a system.

*1) **Model Update Distortion**:* Given the deployment of broadband AirComp, the received aggregated model update at the PS is distorted in two ways. First, the local-model update transmitted by each device may lose some coefficients due to truncated channel inversion in (7). Second, the uncoded aggregated update is directly perturbed by channel noise. There exist a trade-off between these two factors. The sub-carrier/coefficient truncation ratio can be reduced by lowering the truncation threshold in (7). As a result, sub-carriers with small gains are used for transmission, and thus involved in channel inversion, consuming more transmission power. Due to individual devices' fixed power budgets, the magnitude alignment factor, η in (7), has to be reduced. This leads to reduction on the received SNR and more noisy aggregated update received by the PS. For this reason, there exists a trade-off between the truncation ratio of each local-model update and the received SNR. In system design, such a trade-off should be balanced so as to regulate the overall distortion to prevent it from significantly degrading the learning performance. The operating point on this trade-off may also be adjusted along the iterations of the learning process as more accurate estimates of the model updates are needed as the learnign process gradually converges to its optimal value.

*2) **Device Scheduling**:* In a conventional radio-access system, its throughput or link reliability can be enhanced by scheduling cell-interior users at the cost of quality-of-service of cell-edge users. In the context of AirComp FEEL, the penalty of doing so is some loss of data diversity since the data at cell-edge devices cannot be exploited for model training, which can significantly reduce the generalization power of the learned model. To elaborate, due to the required signal-magnitude alignment in AirComp, the received SNR of aggregated model update is dominated by the weakest link among the participating devices. Consequently, including faraway devices with severe path loss can expose model updates to strong noise, and hence potentially slow down convergence and reduce model accuracy. On the other hand, including more devices, which are data sources, means more training data; from this perspective, they may have the opposite effects from the above. Therefore, designing a scheduling scheme for AirComp FEEL needs to balance this trade-off between update quality and data quantity. For example, when the device density is high, the path-loss threshold for selecting contributing devices can be raised and vice versa. On the other hand, mobility can alleviate this issue even when only cell-interior devices are employed. They are mobile and hence change over rounds, which benefits model training by providing data diversity. In the scenario with low mobility, one can also alleviate the issue by alternating cell-edge and cell-interior devices over different rounds [14].

*3) **Coding Against Interference**:* Existing AirComp with uncoded linear analog modulation exposes model training to interference and potential attacks. Most existing works target single-cell systems and overcomes the noise effect by increasing the transmission power. However, in the scenarios of multi-cell networks or multiple coexisting services, the signal-to-interference ratios are independent of power. Besides coping with interference, making FEEL secure is equally important. This motivates the need of coding in AirComp. Possible methods include scrambling signals using pseudo-random spreading codes from spread spectrum or encoding the signals using Shannon-Kotelnikov mappings from joint source-channel coding [104], prior to their transmission. Both coding schemes have the potential of providing the desired property that AirComp remains feasible after coding so long as participating devices apply an identical code (spreading code or Shannon-Kotelnikov mapping) while interference is suppressed by despreading/decoding at the BS.

*4) **Power Control**:* Channel inversion is adopted in typical AirComp to realize magnitude alignment [105]. Its drawbacks are to either exclude devices with weak links from FEEL at the cost of data diversity or consume too much power by inverting such links. In other words, channel-inversion transmission is sub-optimal in terms of minimizing the errors in aggregated gradients/models. Targeting a sensor system with i.i.d. data sources, it was shown in [106] that the optimal power-control policy for error minimization exhibits a threshold based structure: when its channel gain is below a fixed threshold, a device should transmit with full power; otherwise it should adopt channel inversion. Nevertheless, the assumption of i.i.d. data sources does not hold for AirComp FEEL since stochastic gradients or local models of different devices are highly correlated. It is proposed in [107] that information on gradient distribution can be exploited in power control for AirComp FEEL. While this provides significant gains in learning accuracy, the optimal power-control strategy in general remains an open problem.

C. Representative Result

Next, we introduce a concrete example that analyzes the how wireless channels affects the convergence of AirComp FEEL. We consider the implementation of "signSGD" in a broadband system supporting AirComp FEEL. This requires the replacement of analog linear modulation with binary modulation (BPSK). In this case, the decision at the PS depends on the sign of the received signal, and corresponds to an "'over-the-air majority voting" scheme that converts the received aggregated gradient into a binary vector [45]. Given a general loss function, a common metric measuring the level of model convergence is the averaged (aggregated) gradient norm over rounds, denoted as $\bar{G}$. The expectation of $\bar{G}$ for the considered system can be analyzed as a function of given numbers of

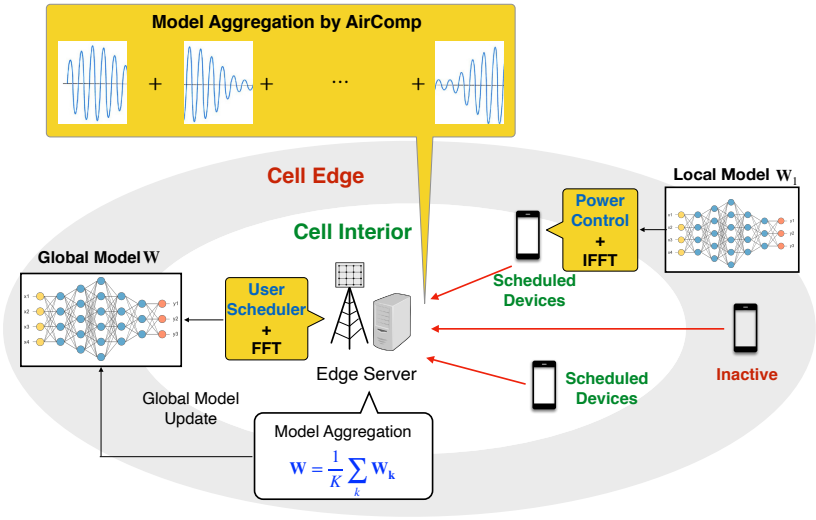


Fig. 5. The AirComp-FEEL system.

rounds and devices, which quantifies the convergence speed. Specifically, it is shown in [108] that

$$\mathsf{E}\left[\bar{G}\right] \leq \frac{a}{\sqrt{N}}\left(f_1 + \frac{1}{\sqrt{K}}f_2 + b\right), \tag{8}$$

where the factors f_1 and f_2/K correspond to the descent using ground-truth gradients and the expected deviation of an aggregated gradient from its ground truth. The two parameters a and b capture the effects of wireless channels. In the ideal case with perfect channels, the parameters take on the values of $a = 1$ and $b = 0$. If the channels are AWGN, they are given as follows:

$$a_{\text{AWGN}} = \frac{1}{1 - \frac{1}{K\sqrt{\text{SNR}}}}, \quad b_{\text{AWGN}} = \frac{f_2}{K\sqrt{\text{SNR}}} \tag{9}$$

where the SNR refers to transmit SNR of a device. One can observe that they converge to their ideal-channel counterparts as the factor $K\sqrt{\text{SNR}}$ grows, where K suppresses noise by aggregation and SNR by increasing signal power. If the channel has fading, then a transmitted local gradient can be truncated as we discussed. Let α denote the probability that a sub-carrier is truncated. Then, the two parameters in this case are given as

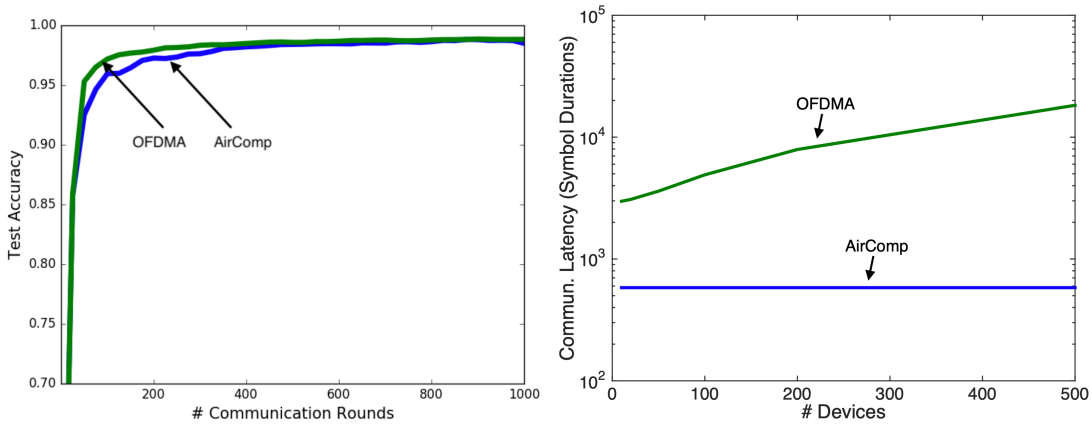
$$a_{\text{FAD}} = \frac{1}{1 - (1-\alpha)^K - \frac{2}{\alpha K\sqrt{\text{SNR}_{\text{av}}}}}, \quad b_{\text{FAD}} = \frac{2f_2}{\alpha K\sqrt{\text{SNR}_{\text{av}}}}. \tag{10}$$

One can see that fading slows down the convergence rate with respect to the AWGN channel since $a_{\text{FAD}} > a_{\text{AWGN}}$ and $b_{\text{FAD}} > b_{\text{AWGN}}$. If the truncation probability $\alpha = 0$, the speeds for both cases are equal since fading is not severe, or the transmission power is sufficiently large to counteract it.

The analysis of convergence speed is useful for estimating the required number of communication rounds for model training. For FEEL, an alternative and perhaps more practical performance metric that can account for multi-access latency is the learning latency (in seconds). It accumulates per-round latency over the total rounds, which is determined by the convergence analysis. On one hand, AirComp achieves a lower model accuracy than the conventional digital orthogonal access due to lack of coding. On the other hand, when there are many devices, AirComp dramatically reduces multi-access latency with respect to the latter. To have an idea on their relative performance, some experimental results from [14] are shown in Fig. 6. The experiment simulates the AirComp-FEEL system in Fig. 5 over broadband channels with 100 edge devices. The task is to train a convolutional neural network using the distributed MNIST data for handwritten digit recognition. The update aggregation is performed by AirComp or OFDMA with adaptive modulation over a broadband channel consisting of 1000 orthogonal sub-channels. FEEL is implemented with local-model uploading. For OFDMA, local-model parameters are quantized using a 16-bit scalar quantizer. The bit sequences of a local model are modulated onto sub-carriers using the classic scheme of adaptive QAM modulation targeting a target bit-error-rate of 10^{-3}. The averge received SNR is set as 10 dB. One can observe from Fig. 6 that under such settings, compared with digital transmission, AirComp can reduce the communication latency by a factor approximately equal to the number of devices without significant loss of the learning accuracy.

D. State-of-the-Art and Research Opportunities

The next-generation IoT is expected to connect tens of billions of edge devices and bring to them computing capabilities and intelligence. Among many others, a specific class of IoT applications has emerged, which requires an edge PS (which can be a BS) to aggregate data distributed at devices with wireless connectivity, termed wireless data aggregation (WDA). Such applications include distributed learning, the topic of this article, as well as vehicle platooning, drone swarm control, and distributed sensing. In such applications, a PS is interested in computing a function of distributed data generated by devices. The applications are either data intensive (e.g., distributed learning) or latency critical (e.g., vehicle platooning). The



(a) Test accuracy (b) Communication latency

Fig. 6. Learning accuracy and latency comparisons between FEEL with AirComp and digital OFDMA [14].

requirements have motivated researchers to develop AirComp to enable efficient WDA over many devices [100]. In the area of FEEL, researchers overcome the communication bottleneck by applying AirComp to implement over-the-air aggregation of local updates.

Analog Compression. Another challenge in AirComp is to enable learning in a broadband communication scenario. When the model updates are transmitted in an uncoded fashion, each iteration of the learning process requires as many channel resources as the model dimension, which can be very large in modern deep learning architectures. For example, architectures used for machine vision applications typically have millions of parameters; well-known AlexNet, ResNet50, and VGG16 architectures have 50, 26, and 138 million parameters, respectively. Models for natural language processing applications typically have much larger networks, with even billions of parameters. Therefore, training such large networks with uncoded transmission would require extremely large bandwidth, often not available at the network edge. This challenge is overcome in [13] by exploiting the sparsity of the model updates. However, note that, sparsification in the case of digital communication of the model updates requires the additional transmission of the index information of the transmitted model parameters from each device, adding a significant additional communication load. In [13], the authors employ random projection of the sparsified model updates at the devices, which allows the devices to significantly reduce the bandwidth requirement without sacrificing the performance. The authors in [109] first analyzed how user selction and transmit power affect the convergence of AirComp based FL and then optimized these wireless factors to improve the performance of AirComp based FL. The work in [110] studied the use of 1-bit compressive sensing (CS) for analog ML model aggregation thus reduce the size of FL parameters transmitted over wireless links. The work in [111] used a Markovian probability model to characterize the temporal structure of the local ML parameter aggregation over a series of learning steps. Based on the Markovian model, the authors developed a turbo message passing algorithm to efficiently recover the desired global ML model from all the historical noisy observations at the PS.

As we discussed previously, researchers have also designed AirComp FEEL systems over multiple-antenna channels [38], [112], [113]. While the beamforming vectors are optimized in [38] to exploit the available multiple antennas for FL, it is shown in [114] that if there are sufficiently many receive antennas at the PS, this can compensate for the lack of channel state information at the transmitter. It is further shown in [114] that, since only the summation of the transmitted symbols needs to be decoded at the receiver, this also reduces the channel state estimation requirements at the receivers, which only needs an estimate of the sum channel gain from the devices to each antenna.

Privacy in FL with AirComp. Another important potential benefit of AirComp in the FL setting is regarding privacy. Even though FL has been proposed as a privacy-sensitive learning paradigm as the devices only transmit their model updates to the PS and the datasets remain localized, it has been shown that the gradient information can reveal significant information about the datasets, called *gradient leakage* [84], [115]. Several works have proposed privacy mechanisms to prevent gradient leakage. In particular, differential privacy (DP) is used as a rigorous privacy measure in this context [116]. A common method to provide DP guarantees is to add noise to data before sharing it with third parties. In the digital implementation of FL, each device can add noise to its local gradient estimate before sharing it with the PS [117], which results in a trade-off between privacy and the accuracy of learning. However, note that, the gradients (or, model updates) in the case of AirComp are received at the PS with additional channel noise. Several recent works have developed privacy-aware AirComp schemes based on this observation. In [118], if the channel noise is not sufficient to satisfy the DP target, some of the devices transmit additional noise, benefiting all the devices. Instead, in [119] and [120], transmit power is adjusted for the same privacy guarantee. The authors in [121] showed that jointly optimizing both wireless aggregation and user sampling can further improve differential privacy. Hence, the authors

designed a private wireless gradient aggregation scheme that relies on the device selection scheme to improve differential privacy. While these works benefit mainly from the presence of channel noise, and depend critically on the perfect channel knowledge at the transmitters, in [122], the authors exploit the anonymity provided by AirComp for privacy, which prevents the PS to detect which devices are participating in each round.

AirComp FEEL is still in its nascent stage. There exist many promising research opportunities. For example, AirComp FEEL can be wirelessly powered to lengthen devices' short battery lives due to intensive computation. As another example, efficient channel feedback based on the AirComp principle can be designed to suppress the excessive feedback overhead when devices are many [38]. Furthermore, the deployment of AirComp FEEL in a multi-cell system exposes learning performance to the effect of inter-cell interference. Quantifying the effect can provide useful guidelines for network designers.

IV. Federated Distillation

Although FL is communication-efficient by nature, it still requires the exchange of large models over the air. Indeed modern DNN architectures often have a large number of model parameters. For instance, GPT-3 model is a state-of-the-art NN architecture for natural language processing (NLP) tasks, and has 175 billion parameters corresponding to over $350\,$GB [123]. Exchanging the sheer amount of deep NN model parameters is costly, hindering frequent communications particularly under limited wireless resources. To address this problem, we introduce FD.

A. Preliminaries

FD only exchanges the models' outputs whose dimensions are much smaller than the model sizes (e.g., 10 classes in the MNIST dataset). For instance, in a classification task, each device runs local iterations while storing the average model output (i.e., logit) per class. Then at a regular interval, these local average outputs are uploaded to the PS aggregating and averaging the local average output across devices per class. Subsequently, the resultant global average outputs are downloaded by each device. Finally, to transfer the downloaded global knowledge into local models, each device runs local iterations with its own loss function in addition to a regularizer measuring the gap between its own prediction output of a training sample and the global average output for the given class of the sample. Such regularization method is called knowledge distillation (KD).

B. State-of-the-Art

While FD was proposed in [124], its effectiveness is not limited to simple classification tasks under a perfectly controlled environment. In [125], FD is extended to an RL application by replacing the aforementioned pre-class averaging step of FD with an averaging operations across neighboring states for an RL task. In [126] and [127], FD is implemented in a wireless fading channel, demonstrating comparable accuracy under channel fluctuations and outages with much less payload sizes compared to FL. The authors in [128] used transfer learning to design a novel FD algorithm which enables edge devices to uniquely design their own ML models. The work in [129] used ensemble distillation for robust model fusion. The designed distillation framework leverages unlabeled data or artificially generated examples to aggregate knowledge from all the received ML models. In [130], the authors applied the new proposed Noise-Free Differential Privacy mechanism into an FD framework thus effectively protecting the privacy of local data with the least sacrifice of the model utility. In [131], a new technique called mix2FLD was proposed whereby local model outputs are uploaded to a PS in the uplink whereas global model parameters are downloaded in the downlink as in FL. To preserve privacy while not compromising accuracy, linearly mixed-up local samples are uploaded, and inversely mixed up across different devices at the PS. For a comprehensive survey on the topic, please refer to [132]

C. Representative Result

To see the effectiveness of FD, we consider the MNIST (hand-written 0-9 images) classification task performed by 10 devices. Figure 8 illustrates the performance of FD for both cases of an IID local dataset and a non-IID dataset whose local data samples are imbalanced across labels. The result shows that for different numbers of devices, FD can always reduce around $10{,}000$x communication payload sizes per communication round compared to FL. Considering both fast convergence and payload size reduction, FD reduces the total communication cost until convergence by over $40{,}000$x compared to FL. Nonetheless, FD still comes at the cost of compromising accuracy, particularly under non-IID data distributions.

D. Summary and Research Opportunities

As shown in the previous section, FD is a very efficient way of training models in a communication-efficient manner, which comes on par with the performance of FL. Preliminary results on MNIST show that FD consumes 10,000x smaller communication cost while FD achieving 95% accuracy of FL under IID datasets. Moreover FD achieves 82% accuracy of FL under non-IID datasets. FD is still in its infancy and several interesting future directions include the co-design of wireless communication and FD as opposed to treating them separately. Another natural extension is to investigate the cost-benefits of model quantization, distillation and their inherent trade-offs.

Beyond supervised learning, FD can be extended towards RL whereby edge nodes collaboratively train a model of their policy and/or value function while taking into account privacy, resource constraints and the inherent nature of wireless connectivity. The learning can be carried out either leveraging a centralized server as in FL or in a totally distributed manner. The challenges in RL are more pertinent in that the environment is non-stationary and the data is non-IID. In both supervised and reinforcement learning setting a theoretical

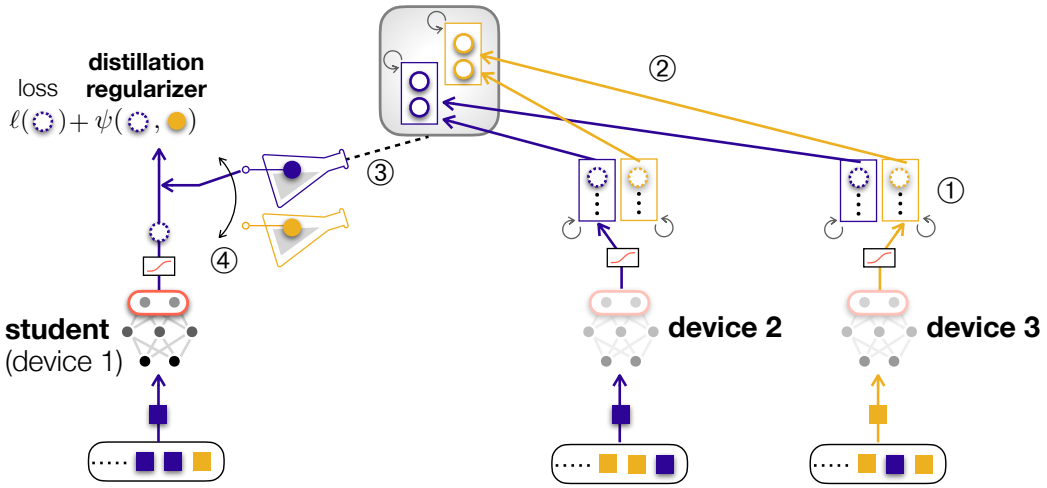
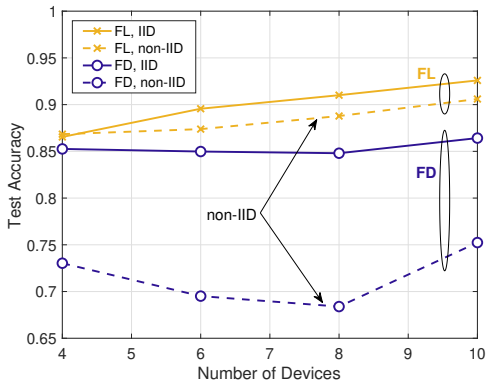
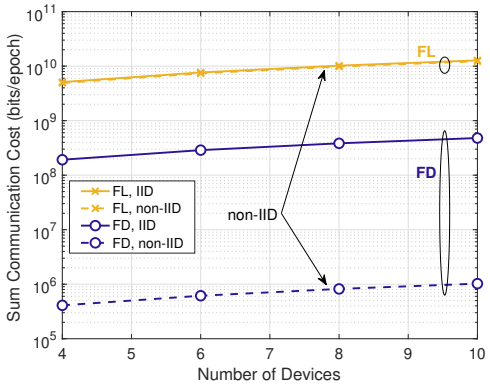


Fig. 7. A schematic illustration of FD with 3 devices and 2 labels in a classification task.



(a) Test accuracy.



(b) Sum communication cost.

Fig. 8. Comparison between FD and FL in terms of (a) test accuracy and (b) sum communication cost of all devices per epoch, under an IID or non-IID MNIST data.

proof of convergence in non-linear networks remains an open problem.

V. Distributed Inference over Wireless Networks

While we have so far focused on the training aspect of ML at the wireless edge, another important component of edge learning is the inference stage. Once an ML model is trained using the available data, this model is then used to make inferences (classification or regression) on new data samples. In standard settings, training is considered to be the most computationally demanding phase of ML problems; and hence, most research has focused on improving the efficiency of distributed training; however, in the case of wireless edge devices and networks, inference is also challenging due to the power and complexity limitations of edge devices, and the latency requirements of the applications. This is particularly the case for inference using complex DNN models, whose dimensions can easily run into hundreds of millions. For example, the popular residual network architecture ResNet-50 for image classification applications consists of 50 convolutional layers, and requires close to 100 megabytes of memory for storage and approximately 4 giga floating point operations (FLOPS) for each image. As an extreme example, the GPT-3 model trained for natural language processing has 175 billion parameters. Implementing such models on edge devices, especially within the time frames required by most edge applications is not feasible due to memory and computational limitations of edge devices. For example, in autonomous vehicles immediate detection of obstacles is critical to avoid accidents, putting stringent latency constraints on the inference task; however, the on-board processing units on a small drone may not be capable of carrying out large DNN inference in a timely manner. Similarly, data rates and processing speeds required in certain applications, such as particle physics experiments [133] or in wireless communications [134] can make online inference extremely challenging. In other scenarios, even when the processing capabilities of the devices or the data volume and latency constraints do not pose a significant challenge, it may not be possible to carry out inference locally if the data to make the inference on is distributed. For example, intelligence for surveillance applications may require images from multiple cameras, or in retrieval tasks, inference may require access to a database available in a remote server [135]. Similarly to federated training, communication becomes indispensable

in such scenarios, and we need to guarantee that distributed inference can still be accomplished within the accuracy and latency constraints of the underlying application. With a few exceptions [135]–[137], so far the physical layer aspects of distributed edge inference have been mostly ignored.

A. Fundamental Limits

From an information theoretic perspective, distributed inference over wireless channels can be treated as a joint source-channel coding version of remote rate-distortion problem [138], [139]. Here, the label to be inferred can be considered as the required reconstruction based on the available data sample. The distortion measure can either be *log loss* in the case of classification, or squared error distortion in the case of regression. Such an interpretation is followed in [140] by further simplifying the inference task as a distributed binary hypothesis testing (HT) problem. Consider, for example, an *observer*, that has independent identically distributed (i.i.d.) samples of a random variable. The observer can communicate over a noisy channel to a remote *decision maker*, which wants to make a decision regarding the underlying probability distribution governing the data samples observed by the observer. Assuming a binary hypothesis testing scenario, e.g., the samples come from one of the two candidate distributions, it is shown in [141] that *separation* is optimal; that is, the simple scheme in which the observer locally performs optimal Neyman-Pearson test and communicates its decision to the tester using the best channel code for the two messages, achieves the best asymptotic error-exponent.

Since we do not impose any computational constraints at the observer, this result is aligned with the intuition that the inference should be made locally at the observer as it has access to all the relevant data, and the implication of this result is that remote inference does not have a substantial impact on the performance, as long as the local decision can be conveyed to the remote decision maker reliably. However, the problem becomes significantly more challenging if both the observer and the decision maker have their own local observations, correlated with each other, and the goal is to make a decision on their joint distribution. In this scenario, since the observer has access only to its own observations, it cannot make a local decision no matter how much processing power it has; instead it must convey some features of its observations to help the decision maker to make the correct decision.

It is known that when the goal of the decision maker is to reconstruct the observation of the observer within some distortion constraint, rather than deciding on their joint distribution, separate source-channel coding is asymptotically optimal [142]; that is, it is optimal for the observer to first compress its observations into as few bits as possible satisfying the distortion constraint, and then to transmit these bits to the decision maker reliably using a capacity-achieving channel code. An interesting question here is whether such a scheme is still optimal when the goal is to make a decision on the joint distribution rather then reconstruction the source samples, such as the case in most ML problems. We note here that, even though hypothesis testing can also be viewed as a rate-distortion problem with a particular distortion metric, it is not an *additive metric* as in standard rate-distortion problems, that is, the distortion between the original source vector and its reconstruction measured by the sum of the distortions between individual elements. It is shown in [140] that the optimality of separation breaks down in the remote hypothesis testing problem. Interestingly, it is also shown in [140] that the optimal error exponent can be achieved by a separation-based scheme for the special case of *testing against independence*; that is, when deciding whether the samples at the observer and the decision maker come from a known distribution or are independent of each other. This result shows that communication and inference cannot be separated even in the asymptotic limit without loss of optimality. On the other hand, how to design such joint schemes is mostly an open area of research.

B. Neural Network Compression and Acceleration

From a practical point of view, since state-of-the-art performance is achieved by DNNs in most practical inference problems, the research has focused on implementing neural network inference on edge devices under the aforementioned constraints on the computational capabilities and memory of the devices, and the available power and bandwidth for communications. A possible approach to solve this problem is model architecture optimization, where the goal is to adjust the size and complexity of DNN architectures to the constraints of the edge device without sacrificing their performance. There are several approaches to achieve this in the literature. A more straightforward approach, similarly to those used for reducing the communication load during training, is to employ parameter pruning and quantization in order to reduce and remove redundant parameters that do not have a significant impact on the performance. It was discovered early on that pruning can reduce the network complexity and help address the overfitting problem [143]–[145]. Today there are many advanced pruning algorithms, and we refer the reader to [146] for a detailed survey. Another effective approach is to impose sparsity constraints during training, through which we directly obtain a sparse network architecture, rather than trying to reduce a complex network to a sparse one through pruning [147], [148]. Network quantization, instead of removing some of the weights, tries to reduce the number of bits required to represent the network weights. In [149], [150], fixed-point representations are employed, and it is shown that very low precision is sufficient not just for inference based on trained networks but also for training them. Some works focused on training neural networks with only a single-bit binary weights [151], [152], showing that DNNs can still perform well, significantly reducing their complexity and memory requirements. In [153], by stochastically binarizing the weights the authors convert multiplications to sign changes, which further simplifies network operations.

DNN compression can also be treated as a standard source compression problem, and vector quantization techniques can be employed for codebook-based compression to reduce the memory requirements. Hashing is used in [154], while [155]

employed vector quantization. Huffman coding is applied in [156] to further reduce the redundancy in quantized network weights.

C. Joint Edge-Device Inference

While offloading data to the edge server for inference is one end of the spectrum, fully local inference using the above compression techniques can be considered as the other end. But, there can be a wide variety of solutions that lie in between, where we benefit from the local computation capabilities of the edge devices, but rather than carrying out full-fledged local inference, we also benefit from the edge servers, and the devices and the edge servers carry out inference tasks in a cooperative manner. A standard approach for edge-device cooperative inference is to split the DNN architecture into two, where the first several layers are carried out on the device, while the remainder are offloaded to the edge server. Such a distributed DNN architecture was first proposed in [157], where a small DNN model is deployed on end devices while a larger NN model is employed in the cloud. For each inference query, the device first rapidly performs local inference using the local model for initial feature extraction, and even completes the inference if the model is confident based on the local features. Otherwise, the end device forwards the result of the local operations to the cloud, which performs further processing and final classification. This approach, by adaptively deciding on the offloading, provides a better use of the local resources and reduces the communication load compared to always offloading to the cloud, but also increases the accuracy compared to fully local inference. Moreover, since only the features that are required for the inference task are offloaded to the edge server, there is an inherent privacy protection as well.

In a parallel work, the neurosurgeon approach in [158] proposed joint computation between the edge device and the edge server by partitioning the layers between the two. By characterizing the per-layer execution time and the amount of data that needs to be conveyed to the edge server at the output of each layer, *neurosurgeon* decides how to divide a complex DNN architecture between the device and the server. This approach is further extended in [159] where the computations in a DNN are modeled as a directed acyclic graph, and the optimal computation scheduling between the edge device and server are studied for a large class of DNN architectures. It is shown in [159] that for generative and autoencoder models, multiple data transfers between the device and the cloud may be required.

It is observed in [158], [159] that, in some DNN architectures, particularly those used for classification tasks, the data size at the output of the initial layers may be even larger than the input size. This would mean that carrying out the initial layers locally at the edge device might increase the communication cost. Lossless compression of the features using Portable Network Graphics (PNG) algorithm [160] is considered in [159]. Further reduction in the communication load can be achieved by using lossy compression. In [161], authors propose applying JPEG compression on the features before transmitting them to the edge server. On the other hand, standard image compression codecs have been designed for visual quality of the reconstructed image; and hence, they may remove high-frequency components of the features that are important for the classification task. In [162], rather than employing standard compression codes, quantized feature maps are compressed using Huffmann coding. Alternatively, in the BottleNet architecture proposed in [163], a learnable feature reduction unit is introduced prior to JPEG compression to make sure only the most relevant features are compressed and forwarded to the edge server.

More recently, in [136], [164], [165] pruning techniques have been combined with DNN splitting to further reduce the computational load on the edge device. Thanks to pruning, more layers can be computed at the edge device within the latency and computational constraints. Pruning also provides a certain level compression by removing some of the less significant features.

D. Joint Edge-Device Inference Over a Wireless Channel

Above approaches abstract out the wireless channel as an error-free ideal bit-pipe, and focus only on the feature compression problem, ignoring the impacts of communication in terms of latency, complexity, or reliability. However, lossy transmission of feature vectors to the edge server over a wireless channel is essentially a *joint source-channel coding (JSCC) problem*, and separation is known to be suboptimal under strict latency constraints imposed by inference problems [166].

While JSCC has been studied extensively in the literature, past work mainly focus on the transmission of image or video sources, following a model-driven approach exploiting particular properties of the underlying source and channel statistics [167], [168]. Recently, an alternative fully data-driven DNN-based scheme, called DeepJSCC, has been introduced [169]. DeepJSCC not only beats state-of-the-art image transmission schemes (e.g., BPG image compression + LDPC channel coding) in many scenarios, particularly in terms of perception sensitive quality measures (e.g., structured similarity index measure, SSIM), but also provides 'graceful degradation' with channel quality, making it attractive for many edge inference applications where the ultra low latency requirements may render channel estimation infeasible. Note also that, these works only consider the latency and the complexity of the operations pertaining to the DNN layers, while ignoring those associated with channel coding/ decoding and modulation, which can be substantial, particularly if we want to operate at a rate close to the capacity of the underlying channel with low probability of error. DeepJSCC significantly reduces the coding/decoding delay compared to conventional digital schemes. Yet another advantage of employing DeepJSCC for edge inference is that, as opposed to conventional digital compression schemes like JPEG of BPG, DeepJSCC has the flexibility to adapt to specific source or channel domains through training. This makes DeepJSCC especially attractive for edge inference as we do not have compression schemes designed for generic feature vectors, whose statistics would change from application to application.

A remote wireless inference problem is considered in [135], where wireless image retrieval is studied. In the scenario considered, the image of a person captured by a remote camera is to be identified within a database available at an edge server. Here, the camera cannot make a local decision as it does not have access to the database. In [135], two approaches are proposed, both employing DNNs for remote inference: a task-oriented DNN-based compression scheme for digital transmission and a DNN-based analog JSCC approach. It is observed that the proposed JSCC approach, which maps the feature vectors directly to channel inputs (no explicit compression or channel coding is carried out), performs significantly better.

In [137], the authors employ JSCC for the joint mobile device- edge server inference problem, and called the new architecture BottleNet++, as this combines the approach in [163] with DeepJSCC. A significant improvement in compression efficiency is achieved by BottleNet++ compared to directly transmitting the compressed feature vectors. In [136], pruning is employed jointly with DeepJSCC, and it is shown that an order of magnitude reduction in required channel bandwidth is possible compared to [137].

VI. Multi-Agent Reinforcement Learning over Wireless Networks

The previous sections introduced the implementation of supervised learning algorithms over wireless networks. Next, we introduce the implementation of RL for wireless network control and optimization.

A. Preliminaries of RL

RL enables the wireless devices to learn the control and management strategies such as resource allocation schemes by interacting with their dynamic wireless environment [17]. Next, we introduce three basic RL algorithms that are generally used for wireless networks.

*1) **Single Agent RL**:* The formal model of a single agent RL can be described as a Markov decision process (MDP) [17]. Hence, the model of a single agent RL consists of four components: agent, state, action, and reward. The agent refers to the device that implements the RL algorithm. The state describes the environment observed by the agent at each time slot. A reward evaluates the immediate effect of an action given a state.

Single agent RL enables the agent to find a policy that maximizes the expected discounted reward while only receiving the immediate reward at each learning step. During the single agent RL training process, the agent first observes its current state, and then performs an action. As a result, the agent receives its immediate reward together with its new state. The immediate reward and new state are used to update the agent's policy. This process will be repeated until the agent finds a policy that can maximize the expected discounted reward.

In wireless networks, single agent RL algorithms can be considered as the centralized algorithms used for network control and optimization. In particular, single RL algorithms that are implemented by a central controller are mainly used for solving non-convex or time dependent optimization problems. For example, one can use single RL algorithms to optimize the trajectory of an unmanned aerial vehicle [170], [171]. However, as the number of mobile devices that are considered by single agent RL increases, the action and state space of the single agent RL will significantly increase thus increasing the training complexity and decreasing the convergence speed. Meanwhile, as the number of the considered devices increases, the overhead of collecting state information of all devices increases which further increases the training complexity of single agent RL. Therefore, it is necessary to design distributed RL that can be jointly implemented by multiple devices.

*2) **Independent Multi-Agent RL**:* Independent multi-agent RL is the simplest MARL algorithm. In the independent MARL, each device implements the single-agent RL individually. In consequence, each device aims to maximize its own expected discounted reward without considering other devices. Given the simple implementation, the agents that perform independent MARL does not need to share any RL information with other devices. Therefore, in wireless networks, independent MARL are generally used for the devices or the BSs that cannot communicate with each other. Since the agents do not share any RL information, independent MARL are not guaranteed to converge and it also cannot find a local optimal solution to maximize the sum expected discounted reward of all agents.

*3) **Collaborative Multi-Agent RL**:* Collaborative MARL requires the agents to share some RL information with other agents. In particular, each agent can share its reward, RL model parameters, action, and state with other agents. For different collaborative RL algorithms, they may share different RL information. For example, the collaborative MARL designed in [172] requires the agents to share their state and action information. In contrast, value decomposition network [173] requires the agents to share their rewards. The training complexity and performance of a collaborative MARL algorithm depends on the information that each agent needs to share. The authors in [174] had compared the training complexity and performance of different collaborative MARL algorithms.

B. State-of-the-Art

Now, we discuss a number of recent works on the use of RL algorithms for network control and optimization. In [175], [176], the authors provided a comprehensive survey for the use of RL for solving wireless communication problems. The authors in [177] proposed a single-agent RL algorithm for optimizing the movement and transmit power of the UAV, phase shifts of the reconfigurable intelligent surfaces (RIS), and the dynamic decoding order. In particular, the authors in [172] designed the recurrent neural network based MARL algorithms for spectrum resource allocation. Using game theory, the designed RL algorithm is proved to converge to a mixed-strategy Nash equilibrium. The authors in [178] designed an independent MARL algorithm for optimizing the spectral efficiencies of BSs and verified that collaborative MARL can achieve better performance than independent MARL. The work in [179] developed a novel MARL algorithm that enables

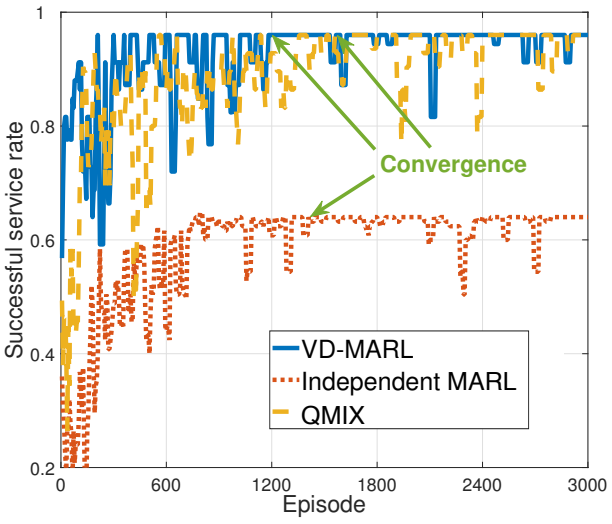


Fig. 9. Convergence of the VD-MARL algorithm [173].

the central controller to use the experience collected from edge devices for training an ML model. The trained ML model will be distributed to edge devices for network performance optimization. The authors in [180] studied the use of cooperative MARL for modulation and demodulation design. In [181], the authors designed a voting-based MARL algorithm that uses a primal-dual algorithm to find the optimal policy for large scale IoT systems. The work in [182] designed a hierarchical federated MARL algorithm for jointly optimizing user association and scheduling. All of the previous works in [172], [175]–[182] are focused on the design of novel RL algorithms for solving wireless communication problems and they did not consider the issues of implementing RL algorithms over resource constrained wireless networks.

Recently, the works in [183]–[187] focused on the design of novel communication efficient RL algorithms. In particular, the authors in [183] proposed a novel policy gradient method for MARL so as to reduce its convergence time. In [184], the authors designed a randomized communication-efficient multi-agent actor-critic algorithm in which a network of multiple agents aim to cooperatively maximize the globally averaged reward through communication with only local neighbors. Meanwhile, the designed algorithm enables each edge device to transmit only two scalar-valued variables for cooperative training at each learning step. The work in [186] proposed a novel model-based RL algorithm that can effectively optimize a policy offline using 10-20 times fewer data than prior works. The authors in [187] propose a deep MARL framework in which agents learn how to schedule themselves, how to encode the messages, and how to select actions based on received messages. Meanwhile, the designed RL framework is capable of deciding which agents should be entitled to broadcasting their messages by learning the importance of each agent's partially observed information.

C. Representative Result

One representative result on the development of MARL for UAV trajectory design can be found in the work [173]. In the considered model, a team of UAVs is dispatched to cooperatively serve several clusters of ground users that have dynamic and unpredictable uplink access demands. The UAVs must cooperatively navigate to maximize coverage of the dynamic requests of the ground users. This trajectory design problem is formulated as an optimization framework whose goal is to find optimal trajectories that maximize the fraction of users served by all UAVs (called successful service rate hereinafter). Traditional optimization algorithms such as branch and bound are not suitable to solve this problem as the successful service rate achieved by each UAV is unpredictable due to the dynamic and unpredictable uplink access demands of ground users. Hence, we designed a novel MARL called VD-MARL that merges the concept of value decomposition network, model agnostic meta-learning, with the policy gradient framework to optimize the trajectories of all UAVs. The proposed MARL algorithm enables each UAV to use the successful service rate achieved by all UAVs to estimate the expected successful service rate achieved by all UAVs over all states thus finding the local optimal trajectories for all UAVs. In particular, implementing VD-MARL, each UAV only needs to share its reward with other UAVs and hence, the overhead of RL information exchange among multiple UAVs significantly reduces.

Fig. 9 shows the convergence of the VD-MARL algorithm. In this figure, we consider three RL algorithms: a) the proposed VD-MARL algorithm, b) the independent MARL algorithm based on actor critic, and c) QMIX in [188]. From Fig. 9, we can see that VD-MARL improve the successful service rate by up to 54% compared to the independent MARL algorithm. This is because the VD-MARL can find a team optimal strategy to maximize the successful service rate of all UAVs. The independent MARL algorithm, however, find a strategy that maximize each UAV's individual successful service rate. This figure also shows that VD-MARL improves the convergence speed by up to 31% compared to the QMIX algorithm. This stems from the fact that the neural network in QMIX used to estimate the estimated future team reward remarkably increases the complexity of QMIX.

D. Research Opportunities

Using MARL for improving wireless network performance requires addressing a number of key problems including:

*1) **Convergence Analysis**:* To analyze the optimality of RL solutions as well as the time and energy used for training RL algorithms, one important problem is to analzye the RL convergence. The existing works have used MDP to analyze the convergence of the single agent RL algorithms and game theory for simple MARL convergence analysis. However, none of these existing works can analzye the convergence of advance MARL algorithms such as QMIX due to complex RL information exchange and neural network updates. Therefore, in this problem, there are several issues including: 1) whether the studied MARL algorithm can find the optimal solution, 2) the number of iterations that MARL needs to converge, 3) how the number of MARL agents affects the convergence, 4) how approximation errors caused by ML models affects the MARL convergence.

*2) **Optimization of Wireless Networks for MARL Implementation**:* In wireless networks, the MARL convergence depends not only on the RL parameters such as the size of ML model but also on the wireless networking factors such as limited number of resource blocks (RBs), imperfect RL parameter transmission, and limited transmit power and computational power of devices. In particular, the number of RBs determines the number of devices that can perform MARL algorithm. Meanwhile, the dynamic wireless channels may cause errors on the transmitted RL parameters. In addition, the limited transmit power and computational power will significantly affect the time used for the RL model update and RL parameter transmission. Therefore, key problems in the implementation of MARL over wireless networks exists in many areas such as 1) optimization of RB allocation and device scheduling for RL parameter transmission, 2) reliable and energy efficient RL parameter transmission, 3) joint optimization of RL training methods and wireless resource allocation for minimizing RL convergence time, 4) coding and decoding method design, and 5) the deployment of advanced wireless techniques such as terahertz and intelligent reflecting surface.

VII. Conclusion

In this paper, we have provided a comprehensive study of the deployment of distributed learning over wireless networks. We have introduced four distributed learning frameworks, namely, FL, FD, distributed inference, and MARL. For each learning framework, we have introduced the motivation for deploying it over wireless networks. Meanwhile, we have presented a detailed literature review, an illustrative example, and future research opportunities for each distributed learning framework. Such an in-depth study on the deployment of distributed learning over wireless networks provides the guidelines for optimizing, designing, and operating distributed learning based wireless communication systems.

References

[1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, Aug. 2019.

[2] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133995–134030, Jul. 2020.

[3] S. Dang, O. Amin, B. Shihada, and M. S. Alouini, "What should 6G be?," *Nature Electronics*, vol. 3, no. 1, pp. 20–29, Jan. 2020.

[4] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. 35, no. 2, pp. 152–159, Mar. 2021.

[5] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sept. 2019.

[6] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, June 2020.

[7] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6G communications: Challenges, methods, and future directions," *China Communications*, vol. 17, no. 9, pp. 105–118, Sept. 2020.

[8] Z. Zhao, C. Feng, H. H. Yang, and X. Luo, "Federated-learning-enabled intelligent fog radio access networks: Fundamental theory, key techniques, and future trends," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 22–28, April 2020.

[9] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, April 2020.

[10] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 2, pp. 1342–1397, Secondquarter 2021.

[11] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[12] M. S. H. Abad, E. Ozfatura, D. Gündüz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.

[13] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, Mar. 2020.

[14] G. Zhu and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[15] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konecny, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proc. Systems and Machine Learning Conference*, Stanford, CA, USA, Feb. 2019.

[17] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, Feb. 2008.

[18] A. Tak and S. Cherkaoui, "Federated edge learning: Design issues and challenges," *IEEE Network*, vol. 35, no. 2, pp. 252–258, Mar. 2021.

[19] C. Shen, J. Xu, S. Zheng, and X. Chen, "Resource rationing for wireless federated learning: Concept, benefits, and challenges," *arXiv preprint arXiv:2104.06990*, 2021.

[20] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.

[22] K. Yang, Y. Shi, Y. Zhou, Z. Yang, L. Fu, and W. Chen, "Federated machine learning for intelligent IoT via reconfigurable intelligent surface," *IEEE Network*, vol. 34, no. 5, pp. 16–22, Sept. 2020.

[23] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S. L. Kim, and M. Debbah, "Communication-efficient and distributed learning over wireless networks: Principles and applications," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 796–819, May 2021.

[24] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs," in *Proc. Annual Conference of the International Speech Communication Association*, Singapore, Singapore, Sept. 2014.

[25] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. Annual Conference of the International Speech Communication Association*, Dresden, Germany, Sept. 2015.

[26] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017.

[27] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sept. 2017.

[28] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. Advances in Neural Information Processing Systems*, pp. 5976–5986. Montreal, Canada, Dec. 2018.

[29] S. U. Stich, J. B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Advances in Neural Information Processing Systems*, pp. 4448–4459. Montreal, Canada, Dec. 2018.

[30] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 2031–2063, Thirdquarter 2020.

[31] Y. Sun, W. Shi, X. Huang, S. Zhou, and Z. Niu, "Edge learning with timeliness constraints: Challenges and solutions," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 27–33, Dec. 2020.

[32] S. Dorner, S. Cammerer, J. Hoydis, and S. T. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[33] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–47, Dec. 2020.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. International Conference on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, USA, April 2017.

[35] V. Smith, C. K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017.

[36] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Advances in Neural Information Processing Systems*, Virtual Conference, Dec. 2020, vol. 33, pp. 3557–3568.

[37] M. Gastpar, "Uncoded transmission is exactly optimal for a simple Gaussian sensor network," *IEEE Transactions on Information Theory*, vol. 54, pp. 2008–2017, Nov. 2008.

[38] G. Zhu and K. Huang, "MIMO over-the-air computation for high-mobility multimodal sensing," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6089–6103, Aug. 2019.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, USA, June 2016.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. International Conference on Learning Representations*, San Diego, California, USA, May 2015.

[41] N. F. Eghlidi and M. Jaggi, "Sparse communication for training deep networks," *arXiv 2009.09271*, 2020.

[42] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. Advances in Neural Information Processing Systems*. Montreal, Canada, Dec. 2018.

[43] E. Ozfatura, K. Ozfatura, and D. Gündüz, "Time-correlated sparsification for communication-efficient federated learning," *arXiv cs.LG 2101.08837*, 2021.

[44] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE International Conference on Neural Networks*, San Francisco, CA, USA, Mar. 1993.

[45] J. Bernstein, Y. X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. International Conference on Machine Learning*, Stockholm, Sweden, Jul. 2018.

[46] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signSGD with majority vote is communication efficient and fault tolerant," in *Proc. International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[47] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes SignSGD and other gradient compression schemes," in *Proc. International Conference on Machine Learning*, Long Beach, CA, USA, Jun. 2019.

[48] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication efficient federated learning," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 118, no. 17, Apr. 2021.

[49] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. International Conference on Artificial Intelligence and Statistics*, Virtual Conference, Apr. 2021, vol. 130, pp. 2350–2358.

[50] S. Caldas, J. Konečny, H B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[51] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, to appear, 2020.

[52] A. Albasyoni, M. Safaryan, L. Condat, and P. Richtárik, "Optimal gradient compression for distributed and federated learning," *arXiv preprint arXiv:2010.03246*, 2020.

[53] X. Dai, X. Yan, K. Zhou, H. Yang, K. K. W. Ng, J. Cheng, and Y. Fan, "Hyper-sphere quantization: Communication-efficient sgd for federated learning," *arXiv preprint arXiv:1911.04655*, 2019.

[54] S. Zheng, C. Shen, and X. Chen, "Design and analysis of uplink and downlink communications for federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2150–2167, July 2021.

[55] A. Abdi, Y. M. Saidutta, and F. Fekri, "Analog compression and communication for federated learning over wireless mac," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications*, Atlanta, GA, USA, May 2020.

[56] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. International Conference on Machine Learning*, Virtual Conference, Jul. 2020.

[57] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017.

[58] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient sgd via gradient quantization and encoding," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017.

[59] S. Horvath, C. Y. Ho, L. Horvath, A. N. Sahu, M. Canini, and P. Richtarik, "Natural compression for distributed deep learning," *arXiv preprint arXiv:1905.10988*, 2019.

[60] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. International Conference on Artificial Intelligence and Statistics*, Palermo, Sicily, Italy, Oct. 2020.

[61] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of federated learning over a noisy downlink," *IEEE Transactions on Wireless Communications*, to appear, 2021.

[62] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Advances in Neural Information Processing Systems*. Long Beach, CA, USA, Dec. 2017.

[63] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[64] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conference on Computer Communications*, Paris, France, 2019.

[65] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, June 2019.

[66] R. Balakrishnan, M. Akdeniz, S. Dhakal, and N. Himayat, "Resource management and fairness for federated learning over wireless edge networks," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications*, Atlanta, GA, USA, May 2020.

[67] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, Jan. 2020.

[68] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[69] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, Jan. 2021.

[70] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.

[71] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, Feb. 2021.

[72] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," *arXiv preprint arXiv:2007.09511*, 2020.

[73] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. of Advances in Neural Information Processing Systems*. Montreal Canada, Dec. 2018.

[74] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019.

[75] R. Kassab and O. Simeone, "Federated generalized bayesian learning via distributed stein variational gradient descent," *arXiv preprint arXiv:2009.06419*, 2020.

[76] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," in *Proc. International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020.

[77] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, USA, Jan. 2019.

[78] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. Advances in Neural Information Processing Systems*, Virtual Conference, Dec. 2020, pp. 21394–21405.

[79] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. Advances in Neural Information Processing Systems*, Virtual Conference, Dec. 2020.

[80] H. Xing, O. Simeone, and S. Bi, "Federated learning over wireless device-to-device networks: Algorithms and convergence analysis," *arXiv preprint arXiv:2101.12704*, 2021.

[81] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. International Conference on Learning Representations*, Virtual Conference, Apr. 2020.

[82] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *Proc. International Conference on Machine Learning*, Virtual Conference, July 2021.

[83] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 48–54, Dec. 2020.

[84] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019.

[85] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, April 2017.

[86] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Proc. International MICCAI Brainlesion Workshop*, Granada, Spain, Sept. 2018.

[87] M. Rojek and R. Daigle, "AI FL for IoT," Presentation at MWC 2019 https://www.slideshare.net/byteLAKE/bytelake-and-lenovo-presenting-federated-learning-at-mwc-2019, 2019, Accessed: 2021-01-17.

[88] F. Díaz González, "FL for time series forecasting using LSTM networks: Exploiting similarities through clustering," Master thesis, KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-254665, 2019, Accessed: 2021-01-17.

[89] S. Ickin, K. Vandikas, and M. Fiedler, "Privacy preserving QoE modeling using collaborative learning," in *Proc. the Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, Los Cabos Mexico, Oct. 2019.

[90] K. Vandikas, S. Ickin, G. Dixit, M. Buisman, and J. Åkeson, "Privacy-aware machine learning with low network footprint," Ericsson Technology Review article https://www.ericsson.com/en/ericsson-technologyreview/archive/2019/privacy-aware-machine-learning, 2019, Accessed: 2021-01-17.

[91] M. Isaksson and K. Norrman, "Secure federated learning in 5G mobile networks," in *Proc. IEEE Global Communications Conference*, Taipei, Taiwan, Dec. 2020.

[92] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," *[Online] Available: https://arxiv.org/abs/1604.00981*.

[93] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. International Conference on Machine Learning*, Sydney, Australia, Aug. 2017.

[94] M. Kamp, L. Adilova, J. Sicking, F. Huger, P. Schlicht, T. Wirtz, and S. Wrobe, "Efficient decentralized deep learning by dynamic model averaging," *[Online] Available: https://arxiv.org/abs/1807.03210*.

[95] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3498–3516, Oct. 2007.

[96] R. Soundararajan and S. Vishwanath, "Communicating linear functions of correlated gaussian sources over a MAC," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1853–1860, Mar. 2012.

[97] G. Mergen and L. Tong, "Type based estimation over multiaccess channels," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 613–626, Feb. 2006.

[98] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3863–3877, Sep. 2013.

[99] L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Over-the-air computation for IoT networks: Computing multiple functions with antenna arrays," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5296–5306, Dec. 2018.

[100] G. Zhu, J. Xu, K. Huang, and S. Cui, "Over-the-air computing for wireless data aggregation in massive IoT," *[Online] Available: https://arxiv.org/abs/2009.02181*.

[101] R. C. Buck, "Approximate complexity and functional representation," *Journal of Mathematical Analysis and Applications*, vol. 70, no. 1, pp. 280–298, July 1979.

[102] G. Arunabha, J. Zhang, J. G. Andrews, and R. Muhamed, *Fundamentals of LTE*, Prentice Hall Communications Engineering and Emerging Technologies Series, 2010.

[103] Y. Shao, D. Gündüz, and S. C. Liew, "Federated edge learning with misaligned over-the-air computation," *arXiv cs.IT:2102.13604*, 2021.

[104] P. A. Floor F. Hekland and T. A. Ramstad, "Shannon-kotelnikov mappings in joint source-channel coding," *IEEE Transactions on Communications*, vol. 57, no. 1, pp. 94–105, Jan. 2009.

[105] X. Cao, G. Zhu, J. Xu, and S. Cui, "Optimized power control for over-the-air federated edge learning," *arXiv preprint arXiv:2011.05587*, 2020.

[106] X. Cao, G. Zhu, J. Xu, and K. Huang, "Optimal power control for over-the-air computation in fading channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7498–7513, Nov. 2020.

[107] N. Zhang and M. Tao, "Gradient statistics aware power control for over-the-air federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5115–5128, Aug. 2021.

[108] G. Zhu, Y. Du, D. Gunduz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 2120–2135, 2021.

[109] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Joint optimization of communications and federated learning over the air," *arXiv preprint arXiv:2104.03490*, 2021.

[110] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "1-bit compressive sensing for efficient federated learning over the air," *arXiv preprint arXiv:2103.16055*, 2021.

[111] D. Fan, X. Yuan, and Y. J. A. Zhang, "Temporal-structure-assisted gradient aggregation for over-the-air federated edge learning," *arXiv preprint arXiv:2103.02270*, 2021.

[112] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, March 2020.

[113] S. Wang, Y. Hong, R. Wang, Q. Hao, Y. C. Wu, and D. W. K. Ng, "Edge federated learning via unit-modulus over-the-air computation," *arXiv preprint arXiv:2101.12051*, 2021.

[114] M. M. Amiri, T. M. Duman, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Collaborative machine learning at the wireless edge with blind transmitters," *IEEE Transactions on Wireless Communications*, to appear, 2021.

[115] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, May 2019.

[116] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, Aug. 2014.

[117] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, Oct. 2016.

[118] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," in *Proc. IEEE International Symposium on Information Theory*, Los Angeles, CA, USA, June 2020.

[119] Y. Koda, K. Yamamoto, T. Nishio, and M. Morikura, "Differentially private aircomp federated learning with power adaptation harnessing receiver noise," in *Proc. IEEE Global Communications Conference*, Taipei, Taiwan, Dec. 2020.

[120] D. Liu and O. Simeone, "Privacy for free: Wireless federated learning via uncoded transmission with adaptive power control," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 170–185, 2021.

[121] M. Seif, W. T. Chang, and R. Tandon, "Privacy amplification for federated learning via user sampling and wireless aggregation," *arXiv preprint arXiv:2103.01953*, 2021.

[122] B. Hasircioglu and D. Gündüz, "Private wireless federated learning with anonymous over-the-air computation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Virtual Conference, June 2021.

[123] T. B. Brown, B. Mann, N. Ryder, and et al., "Language models are few-shot learners," in *Proc. Advances in Neural Information Processing Systems*, Virtual Conference, Dec. 2020.

[124] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," in *Proc. Advances in Neural Information Processing Systems Workshop on Machine Learning on the Phone and other Consumer Devices*, Montreal, Canada, Dec. 2018.

[125] H. Cha, J. Park, H. Kim, M. Bennis, and S. L. Kim, "Federated reinforcement distillation with proxy experience replay memory," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 94–101, July-Aug. 2021.

[126] J. Ahn, O. Simeone, and J. Kang, "Cooperative learning via federated distillation over fading channels," *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Barcelona, Spain*, May 2020.

[127] J. Ahn, O. Simeone, and J. Kang, "Wireless federated distillation for distributed edge learning with heterogeneous data," in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Istanbul, Turkey, Sep. 2019.

[128] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," in *Proc. Advances in Neural Information Processing Systems Workshop on Federated Learning for Data Privacy and Confidentiality*, Vancouver, Canada, Dec. 2019.

[129] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019.

[130] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," in *Proc. International Joint Conference on Artificial Intelligence*, Virtual Conference, Aug. 2021.

[131] S. Oh, J. Park, E. Jeong, H. Kim, M. Bennis, and S. L. Kim, "Mix2FLD: Downlink federated learning after uplink federated distillation with two-way mixup," *IEEE Communications Letters*, vol. 24, no. 10, pp. 2211–2215, Oct. 2021.

[132] H. Seo, J. Park, S. Oh, M. Bennis, and S. L. Kim, "Federated knowledge distillation," *arXiv preprint arXiv:2011.02367*, 2020.

[133] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu, "Fast inference of deep neural networks in FPGAs for particle physics," *Journal of Instrumentation*, vol. 13, no. 7, pp. P07027–P07027, Jul. 2018.

[134] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," *arXiv preprint arXiv:1901.05850*, 2019.

[135] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Wireless image retrieval at the edge," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 89–100, Jan. 2021.

[136] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications*, Virtual Conference, May 2020.

[137] J. Shao and J. Zhang, "BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *Proc. IEEE International Conference on Communications Workshop on Edge Machine Learning for 5G Mobile Networks and Beyond*, Virtual Conference, June 2020.

[138] R. Dobrushin and B. Tsybakov, "Information transmission with additional noise," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 293–304, 1962.

[139] J. Wolf and J. Ziv, "Transmission of noisy information to a noisy receiver with minimum distortion," *IEEE Transactions on Information Theory*, vol. 16, no. 4, pp. 406–411, 1970.

[140] S. Sreekumar and D. Gündüz, "Distributed hypothesis testing over discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2044–2066, Apr. 2020.

[141] S. Sreekumar and D. Gündüz, "Hypothesis testing over a noisy channel," in *Proc. IEEE International Symposium on Information Theory*, Paris, France, Jul. 2019.

[142] N. Merhav and S. Shamai, "On joint source-channel coding for the Wyner-Ziv source and the Gel'fand-Pinsker channel," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2844–2855, Nov. 2003.

[143] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proc. Advances in Neural Information Processing Systems*, Denver, CO, USA, Nov. 1989.

[144] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Proc. Advances in Neural Information Processing Systems*, Denver, CO, USA, Nov. 1990.

[145] B. Hassibi, D. G. Stork, G. Wolff, and T. Watanabe, "Optimal brain surgeon: Extensions and performance comparisons," in *Proc. Advances in Neural Information Processing Systems*, San Francisco, CA, USA, Jul. 1993.

[146] J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey," *arXiv preprint arXiv:2005.04275*, 2020.

[147] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016.

[148] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016.

[149] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. International Conference on Machine Learning*, Lille, France, Jul. 2015.

[150] M. Courbariaux, Y. Bengio, and J. P. David, "Training deep neural networks with low precision multiplications," in *Proc. International Conference on Learning Representations Workshop*, San Diego, CA, USA, May. 2015.

[151] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Advances in Neural Information Processing Systems*, Montreal Canada, Dec. 2015.

[152] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv:1602.02830*, 2016.

[153] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," in *Proc. International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016.

[154] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. International Conference on Machine Learning*, Lille, France, Jul. 2015.

[155] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[156] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. International Conference on Learning Representations*, San Juan, Puerto Rico, May. 2016.

[157] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE International Conference on Distributed Computing Systems*, Atlanta, GA, USA, June 2017.

[158] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems*, Xi'an, China, Apr. 2017.

[159] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[160] "Portable network graphics (PNG)," http://www.libpng.org/pub/png, Accessed: 2019-06-11.

[161] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *Proc. IEEE International Conference on Advanced Video and Signal Based Surveillance*, Auckland, New Zealand, Nov. 2018.

[162] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud

execution," in *Proc. IEEE International Conference on Parallel and Distributed Systems*, Sentosa, Singapore, Dec. 2018.

[163] A. E. Eshratifar, A. Esmaili, and M. Pedram, "Bottlenet: A deep learning architecture for intelligent mobile cloud computing services," in *Proc. IEEE/ACM International Symposium on Low Power Electronics and Design*, Lausanne, Switzerland, July 2019.

[164] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE Conference on Computer Communications Workshops*, Paris, France, Apr. 2019.

[165] J. Shao and J. Zhang, "Communication-computation trade-off in resource-constrained edge inference," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 20–26, Dec. 2020.

[166] A. El Gamal and Y. H. Kim, *Network Information Theory*, Cambridge University Press, 2011.

[167] R. E. van Dyck and D. J. Miller, "Transport of wireless video using separate, concatenated, and joint source-channel coding," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1734–1750, Oct. 1999.

[168] G. Cheung and A. Zakhor, "Joint source/channel coding of scalable video over noisy channels," in *Proc. IEEE International Conference on Image Processing*, Lausanne, Switzerland, Sep. 1996.

[169] D. B. Kurka and D. Gündüz, "DeepJSCC-f: Deep joint source-channel coding of images with feedback," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 178–193, May 2020.

[170] C. You, X. Peng, and R. Zhang, "3D trajectory design for UAV-enabled data harvesting in probabilistic LoS channel," in *Proc. IEEE Global Communications Conference*, Waikoloa, HI, USA, Dec. 2019.

[171] D. Ebrahimi, S. Sharafeddine, P. Ho, and C. Assi, "Autonomous UAV trajectory for localizing ground objects: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, to appear, 2021.

[172] M. Chen, W. Saad, and C. Yin, "Virtual reality over wireless networks: Quality-of-service model and learning-based resource management," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5621–5635, Nov. 2018.

[173] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3177–3192, Oct. 2021.

[174] "Pytorch implementations of the multi-agent reinforcement learning algorithms," https://github.com/starry-sky6688/StarCraft.

[175] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3133–3174, Fourthquarter 2019.

[176] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 2, pp. 1226–1252, Secondquarter 2021.

[177] X. Liu, Y. Liu, and Y. Chen, "Machine learning empowered trajectory and passive beamforming design in UAV-RIS wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2042–2055, July 2020.

[178] M. Bennis, S. M. Perlaza, P. Blasco, Z. Han, and H. V. Poor, "Self-organization in small cell networks: A reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 12, no. 7, pp. 3202–3212, July 2013.

[179] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.

[180] C. De Vrieze, S. Barratt, D. Tsai, and A. Sahai, "Cooperative multi-agent reinforcement learning for low-level wireless communication," *arXiv preprint arXiv:1801.04541*, 2018.

[181] Y. Xu, Z. Deng, M. Wang, W. Xu, A. M. So, and S. Cui, "Voting-based multi-agent reinforcement learning for intelligent IoT," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2681–2693, Feb. 2021.

[182] F. Hu, Y. Deng, and A. H. Aghvami, "Correlation-aware cooperative multigroup broadcast 360 video delivery network: A hierarchical deep reinforcement learning approach," *arXiv preprint arXiv:2010.11347*, 2020.

[183] T. Chen, K. Zhang, G. B. Giannakis, and T. Basar, "Communication-efficient policy gradient methods for distributed reinforcement learning," *IEEE Transactions on Control of Network Systems*, to appear, 2021.

[184] Y. Lin, K. Zhang, Z. Yang, Z. Wang, T. Basar, R. Sandhu, and J. Liu, "A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning," in *Proc. IEEE Conference on Decision and Control*, Nice, France, Dec. 2019.

[185] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, and S. Gu, "Deployment-efficient reinforcement learning via model-based offline optimization," in *Proc. International Conference on Learning Representations*, Virtual Conference, May 2020.

[186] M. Agarwal, B. Ganguly, and V. Aggarwal, "Communication efficient parallel reinforcement learning," in *Proc. Conference on Uncertainty in Artificial Intelligence*, Virtual Conference, July 2021.

[187] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," in *Proc. International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[188] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Conference on Machine Learning*, Stockholmsmässan, Stockholm Sweden, Jul. 2018.