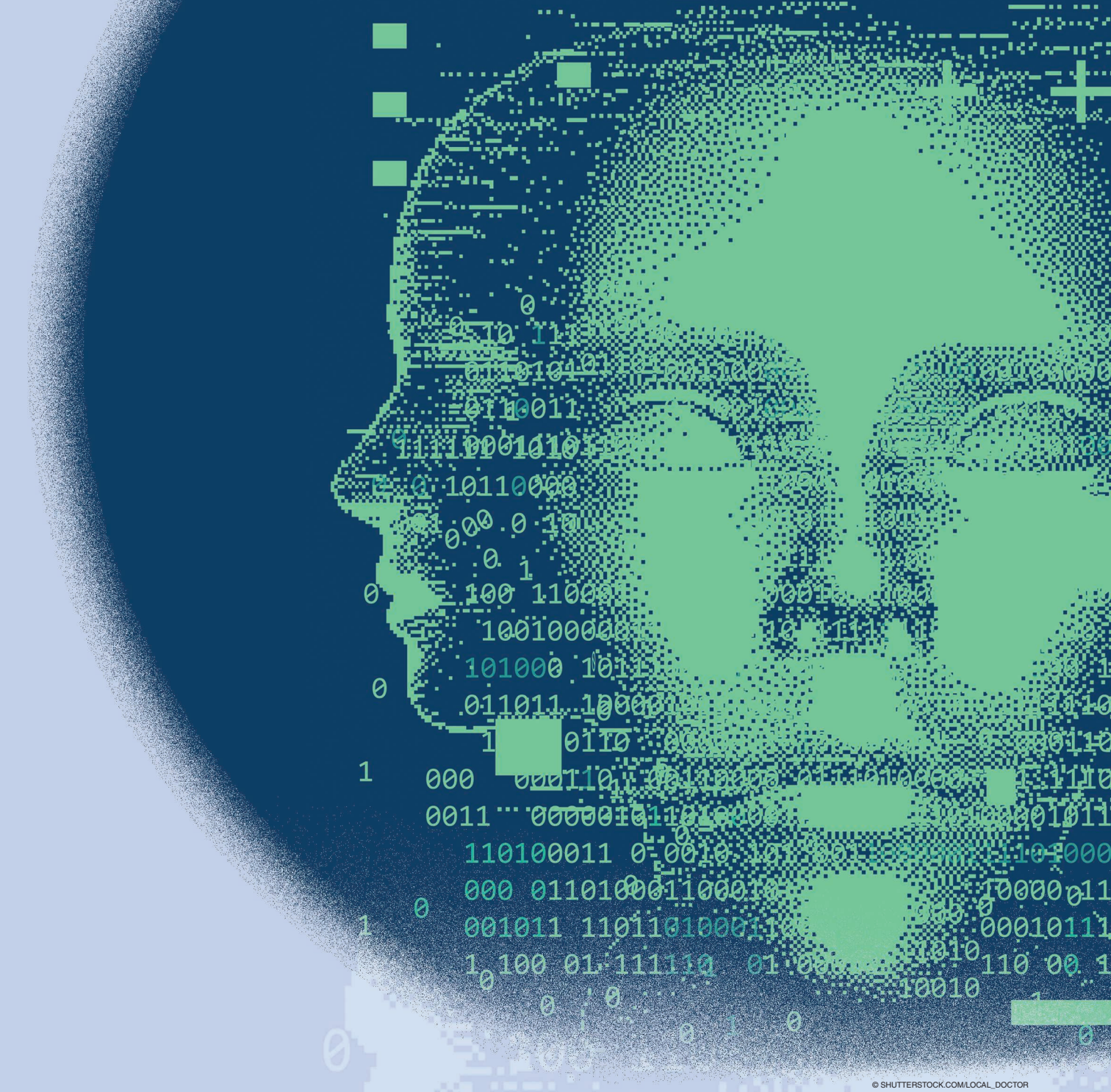


© SHUTTERSTOCK.COM/LOCAL_DOCTOR

Tomer Gafni, Nir Shlezinger, Kobi Cohen, Yonina C. Eldar, and H. Vincent Poor

# Federated Learning

## *A signal processing perspective*

*Digital Object Identifier 10.1109/MSP.2021.3125282*
*Date of current version: 27 April 2022*

The dramatic success of deep learning is largely due to the availability of data. Data samples are often acquired on edge devices, such as smartphones, vehicles, and sensors, and in some cases cannot be shared due to privacy considerations. Federated learning is an emerging machine learning paradigm for training models across multiple edge devices holding local data sets, without explicitly exchanging the data. Learning in a federated manner differs from conventional centralized machine learning and poses several core unique challenges and requirements, which are closely related to classical problems studied in the areas of signal processing and communications. Consequently, dedicated schemes derived from these areas are expected to play an important role in the success of federated learning and the transition of deep learning from the domain of centralized servers to mobile edge devices.

In this article, we provide a unified systematic framework for federated learning in a manner that encapsulates and highlights the main challenges that are natural to address using signal processing tools. We present a formulation for the federated learning paradigm from a signal processing perspective and survey a set of candidate approaches for tackling the concept's unique challenges. We further provide guidelines for the design and adaptation of signal processing and communication methods to facilitate federated learning at a large scale.

## Introduction

Machine learning has led to breakthroughs in various fields, such as natural language processing, computer vision, and speech recognition. Since machine learning methods, and particularly those based on deep neural networks (DNNs), are data driven, their success hinges on vast amounts of training data. These data are commonly generated at edge devices, including mobile phones, sensors, vehicles, and medical devices. In the traditional cloud-centric approach, data collected by mobile devices are uploaded and processed centrally at a cloud-based server or data center. Because data sets, such as images and text messages, often contain private information, uploading them may be undesirable due to privacy and locality concerns. Furthermore, sharing massive data sets can result in a substantial burden on the communication links between the edge devices and the server.

These difficulties can be tackled by exploiting the available computational resources of edge devices via mobile edge computing to carry out training at network edges without having to share data [1]. An emerging paradigm for enabling learning at the edge is federated learning, which features distributed learning with centralized aggregations orchestrated by an edge server (or multiple edge servers) [2]. Federated learning has been the focus of growing research attention over the past few years [3]. In such systems, learning from distributed data and communicating between the server and the devices are two critical and coupled aspects. Their fusion poses many new research challenges that are not encountered in traditional centralized deep learning [4].

Federated learning is based on an iterative training process [2]. At each federated learning iteration, the edge devices train a local model using their possibly private data and transmit the updated model to the central server. The server aggregates the received updates into a single global model and sends its parameters back to the edge devices. Therefore, to implement federated learning, edge devices need only to exchange their trained model parameters, avoiding the need to share their private data. This property of federated learning makes it a promising solution for applications that are comprised of multiple entities that are required to learn from data while operating under strict privacy practices [5]. Federated learning, originally proposed by researchers from Google AI [2], has been applied to improve next-word prediction models in Google's Gboard, allowing a multitude of smartphones to participate in training without requiring users to share their

possibly private text messages. It has since been considered an enabler technology for learning in applications with strict privacy considerations, such as models trained on health-care data; highly distributed technologies, including smart manufacturing systems; and learning over challenging communication networks, such as underwater and unmanned aerial vehicle networks.

Learning in a federated manner is subject to several key challenges that are not encountered in conventional cloud-centric learning [4]. These challenges include the communication bottleneck arising from the need for a large number of users to repeatedly and concurrently communicate their intermediate learned models, which are typically high dimensional, during the learning procedure; the heterogeneity of the participating devices in terms of storage, computation capabilities, and energy states as well as the quantity and distribution of their local data; and privacy and security considerations following from, e.g., the possible presence of malicious users interfering with the training process.

The challenges encountered in federated learning affect the optimization performance and the accuracy of the learned model. They also introduce unique design considerations, such as communication burdens and privacy guarantees, which arise from its distributed nature. Moreover, federated learning adds potentially more hyperparameters to the optimization problem. These include the separate tuning of the aggregation/global model update rule, number of edge devices selected per round, number of local steps per round, configuration of update compression algorithms, and more. These add to the numerous hyperparameters used in conventional deep learning, which, when combined with the aforementioned challenges, make the overall implementation and optimization of federated learning systems a difficult task.

**Data samples are often acquired on edge devices, such as smartphones, vehicles, and sensors, and in some cases cannot be shared due to privacy considerations.**

While the distributed and shared nature of federated learning involves characteristics that are not encountered in conventional centralized deep learning, they are highly related to classical problems studied in the areas of signal processing and communications. Consequently, the derivation of dedicated signal processing schemes is expected to play an important role in the success of federated learning. In fact, a multitude of federated learning-oriented methods based on established concepts in signal processing and communications have been recently proposed. These include the application of compression and quantization techniques to reduce the volume of the messages exchanged in the federated learning procedure [6]–[9]; introducing functional and over-the-air computation tools to facilitate high-throughput federated aggregation over shared wireless channels [10]–[12]; and the derivation of federated learning-aware resource allocation schemes to allow reliable communication between the participating entities during training [13], [14].

In this article, we present leading approaches for facilitating the implementation of federated learning at a large scale using signal processing tools in a tutorial fashion. As opposed to previous tutorials on federated learning, such as [4], which focused on the unique challenges of federated learning and its fundamental differences from centralized deep learning, we focus here on methods for tackling these challenges. To that aim, we discuss how the federated learning paradigm can be viewed from a signal processing perspective, dividing its flow into three main steps: model distribution, local training, and global aggregation. We then focus on the global aggregation step, which involves conveying the local model updates from the users to the central server. We divide this step into three main phases, which are carried out in a sequential fashion: 1) encoding the local model updates at the edge users into messages conveyed to the server; 2) transmitting the model updates and the allocation of the channel resources among the users, which takes into account the statistical relationship between the input and the output of the physical communication channel; and 3) combining (postprocessing) at the server. For each stage, we elaborate on the specific aspects of federated learning that can benefit from tools derived in the signal processing and communication literature with proper adaptation.

We commence by reviewing the federated learning procedure, characterizing its goals and detailing the common federated averaging (FedAvg) optimization algorithm. We then discuss the key challenges of federated learning, elaborating on how they arise from its three-stage operation and particularly from the global aggregation step, which captures the distributed operation of federated learning and its reliance on a multitude of diverse and possibly remote users. For each of the stages that comprise the global aggregation procedure, we systematically identify its interplay with signal processing and communication methods by 1) reviewing the conventional treatment in the federated learning literature, 2) identifying how the paradigm's procedure can be viewed from a signal processing perspective, 3) discussing relevant tools based on this perspective and how they are applied in nonfederated-learning settings, 3) identifying the unique design considerations that one has to account for to adapt these tools for federated learning, and 4) elaborating on existing methods from the recent literature along with detailed examples. The article concludes with a presentation of common guidelines for the derivation of future signal processing schemes for federated learning as well as an overview of some candidate issues for future research.

## Basics in federated learning

In this section, we review the fundamentals of federated learning, identifying the unique role of signal processing in this emerging paradigm. We begin by describing the high-level procedure by which federated learning enables the distributed training of a centralized model. Then, we review conventional optimization mechanisms used in federated learning, after which we discuss the main challenges associated with the

implementation of large-scale federated learning systems. We conclude this section by formulating the three phases of federated learning, which are used in the division of our review of signal processing methods for federated learning in the following sections.

## Federated learning flow

Machine learning systems are data driven; that is, their operations are learned from data. Thus, machine learning requires an algorithm that dictates what to learn from data as well as how to do so, i.e., the learning mechanism. For instance, deep learning relies on highly expressive models based on DNNs, typically trained using variations of stochastic gradient descent (SGD) optimization. In addition, a sufficiently large data set should be provided to the algorithm for learning purposes. Finally, adequate computational resources are necessary to process the data. In conventional centralized machine learning, all these ingredients are available on the same system, which is typically a powerful cloud sever. However, in federated learning, these key ingredients are physically separated, as the data are divided among multiple entities, referred to as *edge users*.

Federated learning deals with the training of a single machine learning model, typically a DNN maintained by a centralized server, without having the users share data. While federated learning can be carried out with multiple servers, we focus our description on settings with a single server, as illustrated in Figure 1. As a result, the two main entities in federated learning systems are the edge users, which have access to local data, and the server that wishes to train a model in light of a given loss measure. We next present the objective and the operation of each of these entities, which are combined into the federated learning procedure.

### Edge users

The data used to train the centralized model are collected by edge users, such as smartphones, wearable devices, and autonomous vehicles. The key differences between federated learning and conventional centralized learning are encapsulated in the properties of the edge devices. First, the edge users are more than just sensing devices; they possess sufficient computational resources to, e.g., locally train a DNN. Furthermore, these devices are capable of interacting with the server and with one another over a communication network. Nonetheless, this communication is carried out over shared rate-limited channels, commonly the wireless cellular infrastructure. Finally, the users are not allowed to share their local data, due to, e.g., privacy constraints. Based on these properties, edge devices use their data to locally train a model and share the updated model with the server.

To mathematically formulate the operation of the edge users, we consider a set of $N$ users, indexed by $\{1, \ldots, N\} \triangleq \mathcal{N}$. Each user of index $i \in \mathcal{N}$ has access to a local data set $\mathcal{D}^i$. Focusing on a supervised setting, this data set consists of $n_i$ labeled pairs $\mathcal{D}^i = \{\boldsymbol{a}_n^i, \boldsymbol{b}_n^i\}_{n=1}^{n_i}$, where $\boldsymbol{a}_n^i$ is the training input and $\boldsymbol{b}_n^i$ is the corresponding label. These data sets can be generated from different distributions and are not necessarily independent and identically distributed (i.i.d.). Each edge device of index $i$ uses its data set $\mathcal{D}^i$ to train a local model comprised of $d$ parameters, represented by the vector $\boldsymbol{\theta}^i \in \mathbb{R}^d$. Training is carried out to minimize a local objective $f(\boldsymbol{\theta}^i; \mathcal{D}^i)$, based on a loss measure $\mathcal{L}(\cdot)$. The local objective for user $i$ is given by

$$f(\boldsymbol{\theta}^i; \{\boldsymbol{a}_n^i, \boldsymbol{b}_n^i\}_{n=1}^{n_i}) = \frac{1}{n_i} \sum_{n=1}^{n_i} \mathcal{L}(\boldsymbol{a}_n^i, \boldsymbol{b}_n^i, \boldsymbol{\theta}^i). \tag{1}$$

Consequently, the objective of user $i$ is to recover the parameters $\boldsymbol{\theta}^i$ that minimize (1); i.e.,

$$\boldsymbol{\theta}^{i^*} = \operatorname*{argmin}_{\boldsymbol{\theta}^i} f(\boldsymbol{\theta}^i; \mathcal{D}^i). \tag{2}$$

The fact that edge devices rely on data collected locally implies that each data set $\mathcal{D}^i$ is typically comprised of a relatively small number of samples, which may be insufficient to train an accurate machine learning model. Specifically, a model parametrized by $\boldsymbol{\theta}^{i^*}$, given in (2), is expected to yield relatively high loss values when applied to data samples that do not appear in $\mathcal{D}^i$ even if it is generated from the same distribution; i.e., the resulting model is likely to fail to generalize. Nonetheless, while each user has access to a limited amount of data, the number of users $N$ is typically very large in federated learning, motivating the users' collaboration in the learning procedure.

### Centralized server

The objective of the server is to utilize the data available at the users' side to train a global model with parameters $\boldsymbol{\theta}$. The objective used for learning $\boldsymbol{\theta}$ is given by

$$F(\boldsymbol{\theta}) = \sum_{i=1}^{N} p_i \cdot f(\boldsymbol{\theta}^i; \mathcal{D}^i), \tag{3}$$

where $p_i \geq 0$ are weighted average coefficients satisfying $\Sigma_{i=1}^N p_i = 1$, typically representing the portion of each user's data set out of the overall training samples; i.e., $p_i = n_i / (\Sigma_{j=1}^N n_j)$.

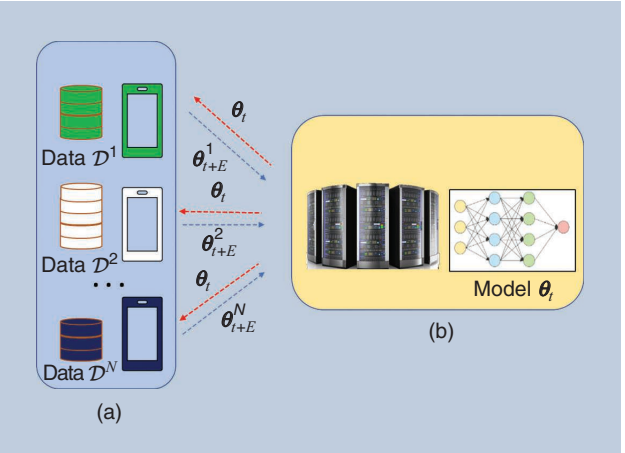



**FIGURE 1.** The federated learning setup, including (a) edge users and (b) a central server.

Consequently, the server aims to solve the following minimization problem:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}). \tag{4}$$

In principle, to recover $\boldsymbol{\theta}^*$, the server must have access to the complete data set $\cup_{i=1}^{N} \mathcal{D}^i$, which is needed to compute the objective in (3). Federated learning allows users to collaboratively train a global model while keeping personal data on their devices, in a manner that is orchestrated by the centralized server, as detailed next.

### Federated learning procedure

To train the global model, federated learning iteratively combines local training based on (2) with centralized aggregation to approach the desired global model in (4). In particular, each round of index $t$ is comprised of the following three main steps:

1) *Model distribution*: The server sends the updated global model $\boldsymbol{\theta}_t$ to the users, and each user sets its current local model to be $\boldsymbol{\theta}_t^i = \boldsymbol{\theta}_t$.
2) *Local training*: Each user $i$ that participates in the training procedure employs its local data to train the local model $\boldsymbol{\theta}_t^i$ into an updated model $\boldsymbol{\theta}_{t+1}^i$ according to (2) via a local optimization algorithm (e.g., several SGD steps).
3) *Global aggregation*: The participating users convey their local updates to the server, which processes the updated local models $\{\boldsymbol{\theta}_{t+1}^i\}$ into the global model $\boldsymbol{\theta}_{t+1}$.

Steps 1–3 are repeated iteratively until convergence. While our description considers supervised learning, the same flow applies for unsupervised settings, where the difference is that the loss measure in (1) is computed based solely on input samples $\{\boldsymbol{a}_n^i\}_{n=1}^{n_i}$. The preceding steps describe the main federated learning flow, which incorporates various algorithms [3]. These methods specify how each of these steps is carried out. The most common mechanism adopted by the majority of federated learning schemes is the FedAvg learning method [2], detailed in the following section.

## FedAvg

FedAvg specializes the federated learning procedure detailed in the previous section by setting the aggregation mapping to implement weighted averaging with the weights $\{p_i\}$ in (3). To mathematically formulate this learning scheme, we focus here on the implementation of FedAvg where the users carry out their local optimization through the SGD algorithm, referred to as *local SGD* [15]. Each user carries out $E > 0$ SGD iterations with minibatch size $B$ prior to their global aggregation and distribution, where each such update is given by

$$\boldsymbol{\theta}_{t+1}^i = \boldsymbol{\theta}_t^i - \eta_t \nabla f(\boldsymbol{\theta}_t^i; \mathcal{D}_t^i). \tag{5}$$

Here, $\eta_t > 0$ is the step size, also referred to as the *learning rate*; $\mathcal{D}_t^i$ is a randomly sampled minibatch with $|\mathcal{D}_t^i| = B$; and $\nabla f(\boldsymbol{\theta}_t, \mathcal{D}_t^i) \in \mathbb{R}^d$ is the stochastic gradient of the objective function with respect to the model parameters evaluated at $\boldsymbol{\theta}_t$.

After $E$ SGD iterations of the form (5), i.e., at iteration $t = kE$, where $k$ is a positive integer, the set of users that participates in the current round, indexed by the set $\mathcal{G}_t \subseteq \mathcal{N}$, conveys its updated models to the server. Typically, the users convey the updates to the model generated in the current round, i.e., $\boldsymbol{\theta}_{kE}^i - \boldsymbol{\theta}_{(k-1)E}^i$, rather than the model weights $\boldsymbol{\theta}_{kE}^i$. Since the server knows $\boldsymbol{\theta}_{(k-1)E}^i$, it can recover $\boldsymbol{\theta}_{kE}^i$ from the difference $\boldsymbol{\theta}_{kE}^i - \boldsymbol{\theta}_{(k-1)E}^i$, while the latter tends to become sparse as convergence is approached. The server then aggregates these local updated models into a new global model $\boldsymbol{\theta}_{t+1}$, where $t = kE$, by computing a weighted average via [15]:

$$\boldsymbol{\theta}_{t+1} = \frac{N}{|\mathcal{G}_t|} \sum_{i \in \mathcal{G}_t} p_i \boldsymbol{\theta}_{t+1}^i. \tag{6}$$

(Alternative averaging-based aggregation mappings have also been considered in FedAvg, such as the inclusion of the previous model of nonparticipating users in the averaging [2]. For consistency, we refer to FedAvg with server-side averaging via (6), as in [15].)

The new global model is broadcast back to the users, which synchronize their local model accordingly. The updating rule of the local SGD algorithm is thus given by

$$\boldsymbol{\theta}_{t+1}^i = \begin{cases} \boldsymbol{\theta}_t^i - \eta_t \nabla f(\boldsymbol{\theta}_t^i; \mathcal{D}_t^i), & \text{if } t \notin \mathcal{I}_T, \\ \dfrac{N}{|\mathcal{G}_t|} \displaystyle\sum_{j \in \mathcal{G}_t} p_j (\boldsymbol{\theta}_t^j - \eta_t \nabla f(\boldsymbol{\theta}_t^j; \mathcal{D}_t^j)), & \text{if } t \in \mathcal{I}_T. \end{cases} \tag{7}$$

Here, $T$ is the number of total steps, and $\mathcal{I}_T \subseteq \{1, \ldots, T\}$ denotes a set of synchronization indices, i.e., the set of integer multiples of $E$. If $\mathcal{I}_T = \{1, \ldots, T\}$, then $E = 1$, and the aggregation is performed at every iteration. If $\mathcal{I}_T = \{T\}$, then aggregation happens only at the end, i.e., $E = T$, which is known as *one-shot averaging*. This iterative procedure converges to the desired optimal global model (4) for strongly convex and smooth objectives with bounded gradients (see "Convergence of Local Stochastic Gradient Descent") at the same asymptotic rate minibatch SGD achieves when carried out in a centralized manner, i.e., when the server has access to the full data set $\cup_{i=1}^{N} \mathcal{D}^i$.

## Challenges

Learning in a federated manner differs from conventional centralized deep learning, where the complete data set is available to the server. This leads to several core challenges that are not encountered in conventional deep learning. These challenges, highlighted in Figure 2, are discussed in the following.

### Communication bottleneck

The repeated exchange of updated models between the users and the server often involves massive transmissions over rate-limited communication channels. This challenge is particularly relevant when the model being trained is a DNN comprised of a large number of trainable parameters as well as for federated

learning carried out over shared and resource-limited wireless networks, e.g., when the users are wireless edge devices. In addition to overloading the communication infrastructure, these repeated transmissions imply that the time required to tune the global model depends not only on the number of training iterations but also on the delay induced by transmitting the model updates at each federated learning iteration. Federated learning networks are potentially composed of a

## Convergence of Local Stochastic Gradient Descent

The unique challenges associated with federated learning affect the convergence analysis of the optimization algorithms and add more hyperparameters. Here, we present the analysis for the convergence rate of the federated averaging (FedAvg) algorithm with local stochastic gradient descent (SGD), which is described in (7), derived from [15]. This bound will serve as a benchmark when discussing the performance of algorithms based on signal processing methods throughout the tutorial.

It is assumed that the local objective functions $\{f(\boldsymbol{\theta};\mathcal{D}^i)\}_{i\in\mathcal{N}}$ are all $L$-smooth, $\mu$-strongly convex; the variance of stochastic gradients in each device is bounded by $\sigma_i^2$; the expected square norm of the stochastic gradients is uniformly bounded by the parameter $G^2$; and $\Gamma$ is defined as the degree of heterogeneity between users. The FedAvg algorithm terminates after $T$ iterations and returns $\boldsymbol{\theta}_T$ as the solution. Choosing $\gamma = \max\{8(L/\mu), E\}$ and learning rate $\eta_t = 2/\mu(\gamma + t)$, FedAvg with full device participation satisfies

$$\mathbb{E}[F(\boldsymbol{\theta}_T)] - F^* \leq \frac{L}{\gamma + T - 1}\left(\frac{2\tilde{B}}{\mu^2} + \frac{\gamma}{2}\mathbb{E}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2\right), \quad \text{(S1)}$$

where $\tilde{B} = \Sigma_{i=1}^N p_i^2 \sigma_i^2 + 6L\Gamma + 8(E-1)^2 G^2$.

The main insight is that FedAvg converges to the global optimum at a rate of $\mathcal{O}(1/T)$, which is the same asymptotic rate minibatch SGD achieves when carried out in a centralized manner. The parameter $E$ (the number of local SGD iterations prior to the aggregation) controls the convergence rate. Setting $E$ to be very small may lead to a high communication cost. On the other hand, if $E$ is large, then $\boldsymbol{\theta}_t^i$ can converge to the minimizer of $f(\boldsymbol{\theta};\mathcal{D}^i)$. It is noted that FedAvg requires diminishing learning rates for convergence.

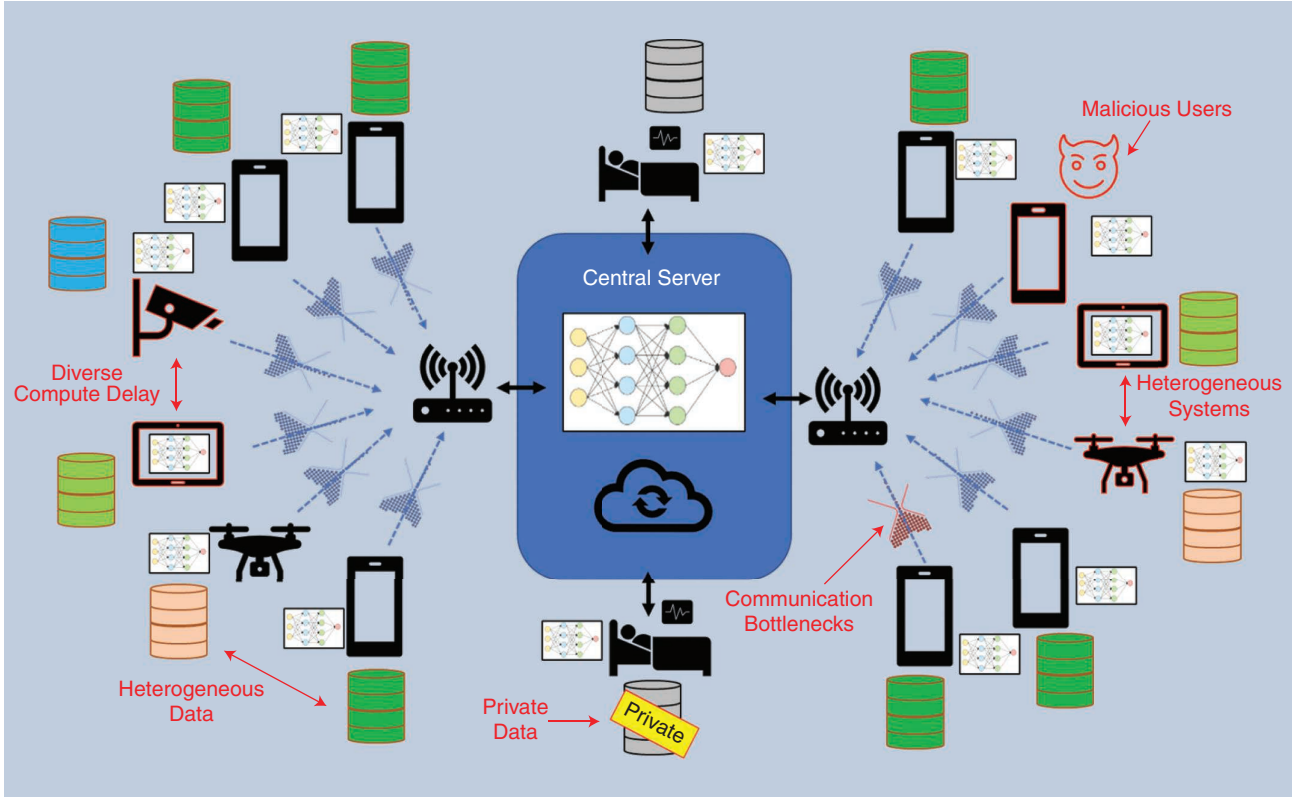



**FIGURE 2.** A federated learning system with various users, highlighting some of the core challenges, including communication bottlenecks, the statistical heterogeneity of the data, heterogeneity in different user devices, diverse hardware and computational delays, the need to preserve privacy with respect to data, and the possible presence of malicious users.

massive number of energy-constrained edge users communicating over wireless media. Considering the limited capacity of wireless channels, the communication of the model updates can be slower than local computation by orders of magnitude. Hence, the communication bottleneck directly affects the training time of global models trained in a federated manner, which, in turn, may degrade their resulting accuracy.

### Statistical heterogeneity

Statistical heterogeneity implies that the data generating distributions vary among different sets of users. This is typically the case in federated learning, as the data available at each user device are likely to be personalized toward a specific user. Statistical heterogeneity implies that when training several instances of a model on multiple edge devices, each instance may be biased. As a result, the conventional strategy based on averaging the trained updates may not reflect the intended usage of the global model in inference.

### System heterogeneity

Federated learning involves a multitude of user devices whose behavior and availability can be quite diverse. This form of heterogeneity, referred to as *behavior heterogeneity*, can affect a learning process that relies on a device status. For instance, one may design the federated learning mechanism to involve portable user devices only when they are idle, charging, or connected to an unmetered network, such as Wi-Fi. As a result, participants may be unreliable and can drop out at any time, leading to uneven participation over the training group. Moreover, as opposed to centralized learning, where training is carried out using a computationally powerful server, the participating devices in federated learning vary in terms of hardware specifications, including computation and energy resources. For instance, a federated learning network may involve user devices ranging from sophisticated autonomous vehicles to hardware-limited Internet-of-Things devices. This implies that the duration of the local training procedure can vary considerably among users, resulting in computation bottlenecks and straggler effects that delay the overall convergence of the learned model.

### Privacy and security

One of the key motivations for federated learning is to help preserve privacy with respect to users' data. In that sense, federated learning has distinct privacy advantages compared to centralized training on persistent data, as the information conveyed from the users is not the local data themselves but the model updates obtained using the data. However, one must also consider that information can be inferred from the learning process and be traced back to its source in the resulting trained model. For instance, gradients are known to be leaky and may reveal the data used for computing them [16]. Furthermore, since the users are outside the control of the server, they may be unreliable or even malicious, giving rise to security issues. Such users send the server arbitrary or tainted model updates, affecting the trained model while exploiting the interactiveness of the process to have the opportunity to adapt.

## Signal processing stages in federated learning

As detailed in the preceding, the federated learning procedure is comprised of three main steps: model distribution, local training, and global aggregation. Nonetheless, the unique challenges of federated learning discussed in the previous section are mostly associated with the last step, global aggregation. In particular, the distribution step involves the broadcasting of the global model from the server to the users and thus is far less affected by privacy, security, and heterogeneity considerations compared to the transmission from the users to the server. Furthermore, this broadcasting is carried out over the downlink channel, which is typically less limited in terms of throughput compared to the uplink channel, and therefore is less affected by communication limitations than the aggregation stage is. The local training step, which yields the updated model based on the previous model and the data, is typically carried out using conventional optimizers, such as SGD and its variants. Consequently, the federated learning step in which the most challenges arise is global aggregation.

Global aggregation can be divided into three main stages, and each can be treated as a signal processing and/or communication task. These stages are 1) the processing and encoding of the local training outcome at the edge users, 2) the transmission of these outcomes over shared wireless channels, and 3) the processing and combining of the received signals at the server as part of the global learning procedure. An illustration of the federated learning procedure, including this division into stages, appears in Figure 3.

### Processing and encoding

The first stage focuses on processing the local outcomes of the training procedure at the users' side before transmission. Such processing is essential to be able to convey the model updates to the server over a rate-limited, shared link in a reliable and privacy-preserving manner. Given the local model for user $i$ at time $t$, $\boldsymbol{\theta}_t^i$, the local processing is the mapping

$$\boldsymbol{s}_t^i = \boldsymbol{\phi}_i(\boldsymbol{\theta}_t^i), \tag{8}$$

where $\boldsymbol{s}_t^i$ is the processed model update used to form the channel input in the transmission stage, as demonstrated in Figure 3.

The mapping (8) should be designed to address the necessity to encode and compress the model updates. As we discuss in the "Local Updates Processing and Encoding" section, which is dedicated to methods associated with this stage, this can be achieved by setting $\boldsymbol{\phi}_i(\cdot)$ to implement the stochastic quantization and sparsification of the model updates for the reduction of the communication load. Furthermore, $\boldsymbol{\phi}_i(\cdot)$ can be designed to boost privacy preservation with respect to the data, via, e.g., inducing local noise perturbations relying on the algorithmic foundations of differential privacy.

### Uplink transmission

The processed updates $\boldsymbol{s}_t^i$ are transmitted to the server for global updating, typically over a wireless channel. To that end,

each user must form its corresponding channel input, denoted $\boldsymbol{x}_t^i$, to convey $\boldsymbol{s}_t^i$ via the mapping

$$\boldsymbol{x}_t^i = \varphi_i(\boldsymbol{s}_t^i). \tag{9}$$

The statistical relationship between the channel input $\boldsymbol{x}_t^i$ and the output obtained at the server side $\boldsymbol{y}_t$ is determined by the channel characteristics encapsulated by the conditional distribution $\mathbb{P}(\boldsymbol{y}_t|\{\boldsymbol{x}_t^i\}_i)$, as shown in Figure 3. That is, the server receives a noisy version of the channel inputs transmitted by the users, where the transmission mapping and the characteristics of the channel dictate the relationship between the channel output $\boldsymbol{y}_t$ observed by the server and the encoded model updates $\boldsymbol{s}_t^i$.

As discussed in detail in the following, methods related to the transmission phase typically focus on federated learning carried out over shared wireless channels. Here, the aim is to allow high-throughput and reliable communication in a manner that does not introduce notable delays to the overall learning procedure. This requires consideration of two main aspects. The first is the allocation of the resources of the channel in terms of bandwidth and transmission time. A special case of channel resources division is user selection and scheduling, determining the set of participating users in the current round $\mathcal{G}_t$. Furthermore, one can exploit the shared nature of the wireless media as a form of over-the-air functional computation, allowing the users to exploit the full resources while benefiting from the resulting interference.

## Global combining

The server uses its observed channel output $\boldsymbol{y}_t$, which contains the information about $\{\boldsymbol{\theta}_t^i\}_{i=1}^N$, to update the global model. This operation, carried out by the server, can be represented by the following mapping:

$$\boldsymbol{\theta}_t = \Phi(\boldsymbol{y}_t). \tag{10}$$

Here, the server uses the channel output, which contains information regarding the individual model

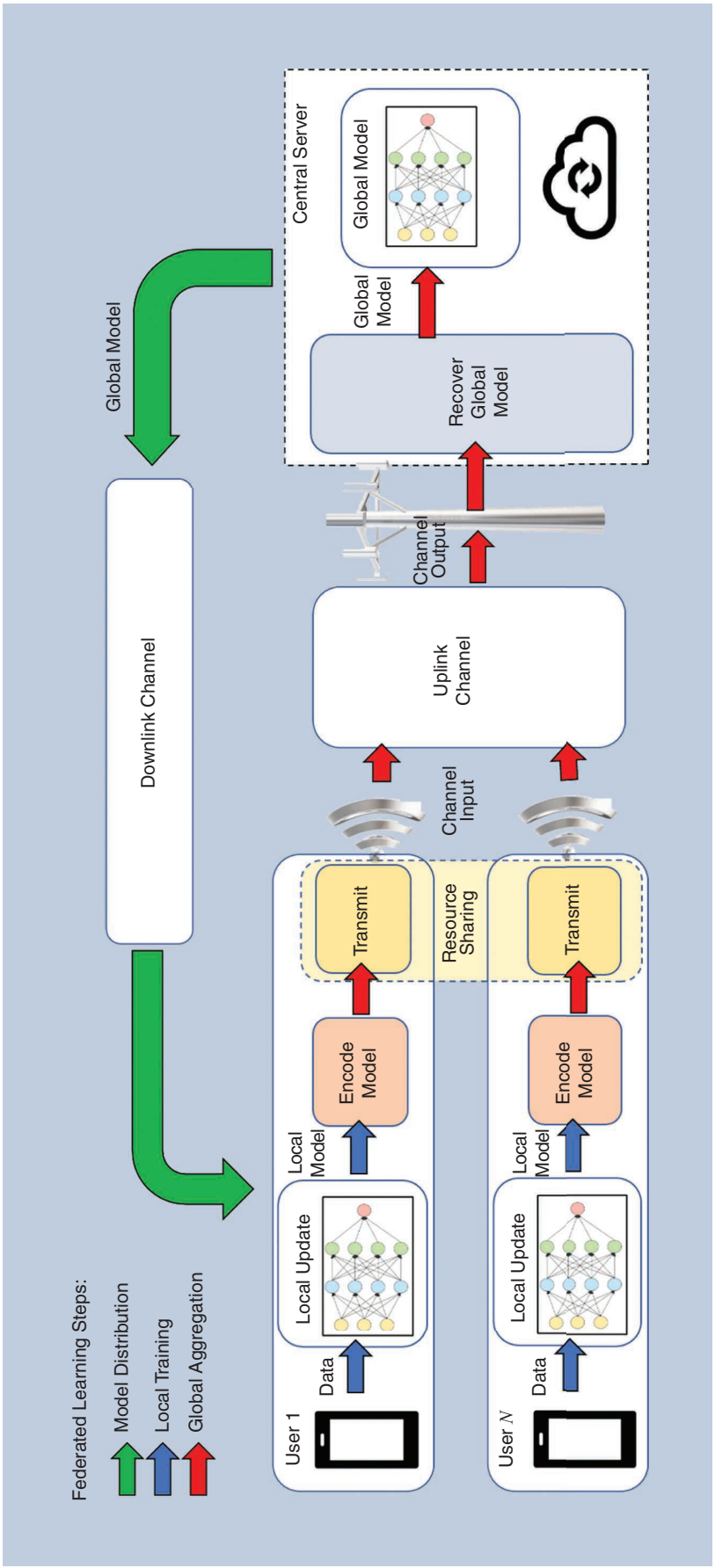



**FIGURE 3.** The federated learning procedure divided into the three main steps of model distribution, local training, and global aggregation. The latter is further represented as a three-stage process, involving the encoding of the model updates, their transmission to the server, and their combination into a global representation.

updates, to estimate a combined global model, as illustrated in Figure 3. As we discuss in the "Global Combining" section, this mapping can also be utilized to compensate for noise and fading induced by the channel, improve the accuracy of the global model during inference by mitigating the effect of statistical heterogeneity, and overcome the possible harmful effect of malicious devices using Byzantine–robust aggregation.

## Local updates processing and encoding

The first stage in the global aggregation step of federated learning focuses on the processing of the local outcomes of the training procedure at the users' side before the results are transmitted. Here, we describe conventional encoding strategies and discuss how this operation can be viewed from a signal processing perspective, briefly reviewing related methods applied in a nonlearning setting. We then discuss the strategies' adaptation to encoding the local updates in federated learning by highlighting specific design considerations and reviewing some candidate techniques from the recent literature.

Given the local model for user $i$ at time instance $t$, $\boldsymbol{\theta}_t^i$, the local encoding process is given by the mapping $\boldsymbol{s}_t^i = \boldsymbol{\phi}_i(\boldsymbol{\theta}_t^i)$ in (8). Because the server is interested in aggregating the model $\boldsymbol{\theta}_t^i$ and not in the encoded symbols $\boldsymbol{s}_t^i$, the encoding procedure should be combined with an appropriate joint decoding process. Since the local data sets possibly differ from one another in their size and distribution, each user will in general have an individual mapping $\boldsymbol{\phi}_i(\cdot)$, induced from its unique characteristics and based on the underlying task.

### *Conventional encoding*

The straightforward implementation of the FedAvg algorithm requires the users to send the model updates to the server. In this case, the code word is merely the difference between the last distributed global model, which is available to the server and identical among all users, and the current locally trained one; i.e.,

$$\boldsymbol{\phi}_i(\boldsymbol{\theta}_t^i) = \boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i. \tag{11}$$

This conventional encoding does not account for any knowledge one may possess about the model updates and their structure. Furthermore, the full model update vector in (11) is of the same dimensionality as the model itself, implying that communicating it to the server is likely to produce a notable communication bottleneck. In addition, one of the main motivating factors behind federated learning is that the local data at each user do not leave the local device, as only gradients or the model update computations from each user are shared with the server. However, even exchanging gradients in a raw form can leak information [16]. Finally, (11) is agnostic to the federated learning task, as the server averages the updates from the users, and this aggregation procedure can mitigate noise in the updates induced by encoding techniques.

### *Signal processing perspective*

The mapping $\boldsymbol{\theta}_t^i \mapsto \boldsymbol{s}_t^i$, carried out at multiple user devices in each round, is, in fact, a form of distributed source coding [17]. The main objective in the source coding literature is compression; i.e., the goal is to map $\boldsymbol{\theta}_t^i$ into a compact representation from which the original vector can be recovered with minimal errors. Federated learning is also concerned with recovering a unified accurate model from these representations, typically via averaging, as well as helping preserve privacy with respect to the local data sets. This gives rise to learning-aware encoding schemes, which can be divided into uplink compression and privacy preservation, as detailed in the following.

### *Uplink compression*

Uplink compression is the procedure of encoding the model updates, having the code word conveyed to the server being comprised of fewer (and preferably many fewer) bits compared to those used for representing the original model. Compression is necessary to reduce the uplink communication cost, thus addressing the communication bottleneck, which is among the core challenges of federated learning.

#### Source coding for compression

Compression is the objective of traditional source coding methods. Source coding techniques are typically divided into lossless and lossy compression. Lossless compression, such as entropy coding, converts discrete quantities into a reduced-bit representation in a reversible manner. Lossy compression, including quantization, can also be applied to continuous-valued quantities and yields a compact digital representation from which the original input can be recovered only up to some error. Since model parameters trained in a federated manner, e.g., the weights of DNNs, are typically treated as continuous-valued quantities during training, we henceforth focus on lossy compression methods. Furthermore, lossy compression facilitates achieving improved compression rates compared to lossless compression for discrete-valued weights (though at the cost of possible distortion upon recovery). Finally, it is common to combine lossy and lossless compression, e.g., first quantizing and then applying entropy coding, as in entropy-constrained quantization.

Source coding techniques also extend to distributed setups, where multiple users encode correlated local sources (e.g., vectors) such that the resulting representations are compact in terms of the number of bits and can be jointly recovered using a single decoder [17]. Existing distributed source coding methods, such as lossless Slepian–Wolf coding and lossy Wyner–Ziv coding, typically require the encoders to have some knowledge of the joint statistics of the sources available at each of the users. Distributed source coding methods enable the users to exploit this correlation among the sources to further compress them without compromising the ability to reconstruct the sources from the compressed representations.

### Design considerations

The main distinction between uplink compression for federated learning and conventional source coding follows from the specific characteristics of the learning setup. In particular, compression methods in federated learning should account for the following considerations:

1) The model updates in federated learning are characterized by the lack of a unified statistical model, and the joint distribution of the model updates is not available.
2) The model compression procedure is a part of the global aggregation stage, and the compressed updates are to be transmitted through the uplink channel.
3) The encoding procedure should be combined with an appropriate joint decoding process, as the server is interested in aggregating the models $\{\boldsymbol{\theta}_t^i\}$, not the encoded symbols $\{\boldsymbol{s}_t^i\}$.

### Uplink compression methods

While the local encoding stage can be treated as a source coding setup, distributed source coding techniques are typically not suitable for compressing the model updates. This follows mainly from the lack of a known statistical characterization of the joint distribution of the model updates, which is typically exploited by distributed source coding methods. Furthermore, the fact that in federated learning different users are often randomly selected to participate in each round (see the "Learning-Aware Resource Allocation" section) and that aggregation often involves truncating some of the model updates (see the "Security-Enhanced Federated Combining" section) makes the application of distributed source coding techniques, which relies on the joint decoding of all the coded representations, quite challenging to implement. Therefore, uplink compression in federated learning is carried out by each user individually in a disjoint manner with the remaining users, and (nondistributed) lossy source coding is typically employed to achieve highly compressed representation.

Lossy compression can cause losses of information and therefore leads to accuracy degradation in the optimization task. However, the fact that the updates are compressed for a specific task, i.e., to obtain the global model by, for example, FedAvg, implies that federated learning with a bit constraints setup can be treated as a task-based compression scenario. This task is accounted for in the selection of the compression scheme, using one for which the error term vanishes by averaging regardless of the values of $\{\boldsymbol{\theta}_t^i\}$. In particular, uplink compression methods in federated learning can be divided to two general approaches:

1) *Sparsification*: The model updates $\boldsymbol{\theta}_t^i - {}_{t-E}^{i}$ are encoded by preserving only a subset of their entries. Sparsification can be implemented based on the value of the model updates, e.g., by nullifying the model updates of lesser magnitude [19] or by encoding each layer into a low-rank approximation [20]. The rationale here is that nullifying the weak parameters is expected to have only a minor effect on the accuracy of the model. Alternatively, one may sparsify the model updates in an arbitrary fashion by using randomized subsampling and sparsifying masks [20]. The latter strategy relies on the fact that when the masks are randomized in an i.i.d. manner among the users, the global model aggregated via FedAvg will hardly be affected by this sparsification when the number of participating users is sufficiently large.
2) *Quantization*: Each parameter (or set of parameters) is mapped into a finite-bit representation. For example, one-bit quantization can be implemented by taking the sign of each model update entry. In fact, when the model updates are the stochastic gradients, that is, when each user implements only $E = 1$ iteration in the local training, one can still guarantee the convergence of the learned model with this strategy, referred to as *sign SGD* [21]. Nonetheless, this crude quantization can notably reduce the convergence rate and the accuracy of the learned model after a finite number of iterations.

   Quantization schemes studied in the areas of compressionand analog-to-digital-conversion often require knowledge of the distribution of their input to determine which values are mapped to which discrete representation. This limits their application in federated learning setups. However, by exploiting dithered (randomized) quantization techniques, federated learning of accurate models can be achieved regardless of the distribution of the model updates, e.g., [7]. This follows since the distortion induced by dithered quantization is statistically uncorrelated with the model updates, and thus its effect is mitigated by the averaging operation carried out by the server. Furthermore, dithered quantization can greatly benefit from having a source of common randomness between the users and the server [8], which is a feasible requirement in federated learning, and hence a shared random number generator allows for implementing subtractive dithered quantization [22], as illustrated in Figure 4.

An example of compression for federated learning in the form of universal dithered quantization is given in "Dithered Quantization for Federated Learning." Both approaches can be followed by a lossless source code, exploiting the structures induced by sparification and coarse discretization to further compress the model updates. Sparsified model parameters can also be compressed using simple linear projections, from which the sparse vector can be recovered using compressed sensing mechanisms [23], [24].

### *Privacy preservation*

The large-scale collection of sensitive data entails risks to the privacy of individual users. Although each local update does not explicitly contain a user's data, local updates may be harnessed to reveal the data used for computing them [16]. In general, there are several potential adversaries from which privacy should be protected in federated learning: a server that may attempt to infer private information by using all received model updates, malicious users that can deviate from the protocol execution to achieve information about data held

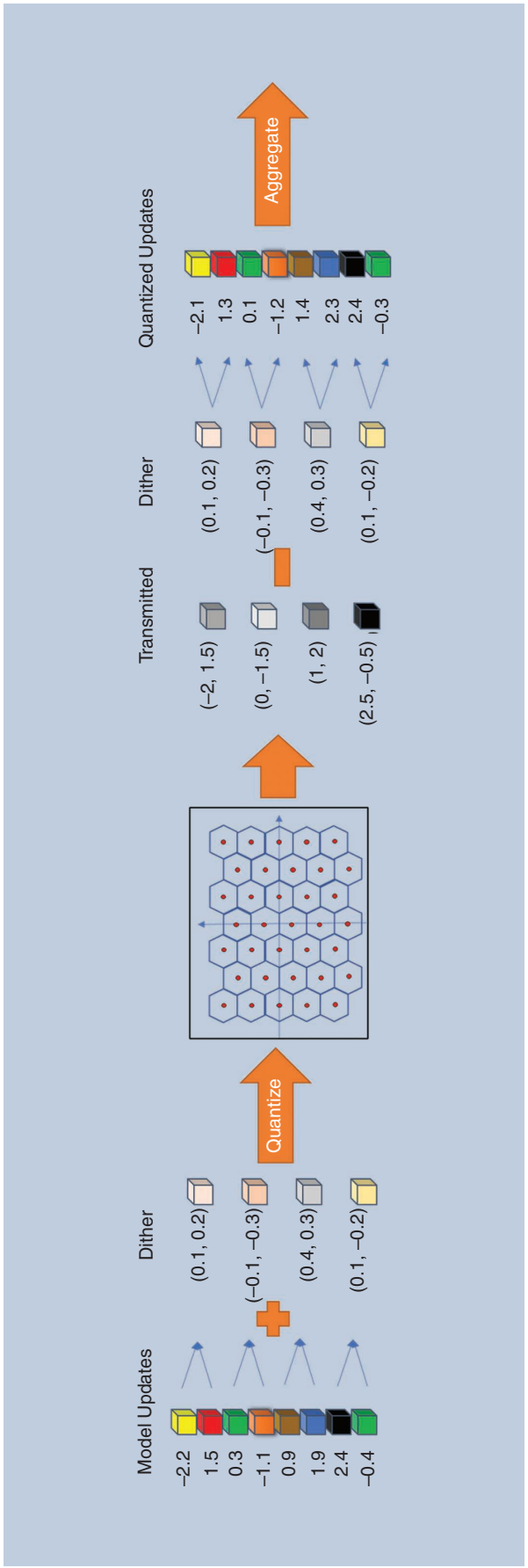



**FIGURE 4.** Universal quantization via random lattices.

by honest ones (see the "Security-Enhanced Federated Combining" section), and external adversaries that try to access private data.

## Encoding for privacy

A natural strategy to encode the model updates to help preserve privacy is to encrypt them. Encrypting the model updates implies that they cannot be recovered and thus cannot be traced back to the data by external adversaries. To help preserve privacy while also combining the encrypted model updates into a global model, one can use secure distributed coding mechanisms, also known as *multiparty encryption*. In particular, the family of additively homomorphic multiparty encryption schemes facilitates performing additive calculations, e.g., averaging-based aggregation, on the encrypted data without having to first decrypt them [25]. The result of the computation is an encrypted form such that when it is decrypted, the output is the same as if the operation had been performed on the unencrypted data.

Multiparty encryption methods require the participating entities to share a secret key. An alternative approach to implement secure encoding without key sharing is by randomly obscuring the source via, e.g., artificial noise. This strategy is widely used in physical layer security mechanisms [18], which are well studied in the signal processing and communication literature. Here, the noise is typically designed to achieve a vanishing mutual information criterion, known as *weak secrecy*, at the adversary side, i.e., after it has been transmitted. Encoding by artificial noise can achieve privacy guarantees that are more powerful than weak secrecy and invariant of transmission, such as differential privacy [26]. The differential privacy level represents how much the algorithm output changed when modifying a single data sample and thus reflects the ability to retrace a data sample from the outcome of the training procedure [the exact mathematical formulation of this notion of $(\epsilon, \delta)$ differential privacy is given in "Privacy Boosting by Noising Before Model Aggregation"]. Encoding by artificial noise can guarantee achieving a predefined level of differential privacy, where the noise should be proportional to the sensitivity of the output, i.e., the maximum change of the output due to the inclusion of a single data instance.

## Design considerations

Encoding for privacy allows the users to guarantee, to a quantifiable degree, that the data used in training their local models can be recovered neither by the server nor external adversaries. Nonetheless, federated learning brings forth the following considerations, which should be accounted for in the design of privacy-preserving encoding methods:

1) Federated learning is not a one-shot operation but is carried out over multiple rounds, typically with different users participating in each round, due to device selection and dropout. Privacy-preserving encoding methods thus must be fully robust to users dropping out at any point. For instance, the use of secure key distribution protocols requires a predefined threshold of users to decrypt the global model,

implying that dropout may result in an inability to decrypt the global model.

2) It is desirable not to substantially increase the communication overload for privacy preservation.
3) The model updates $\boldsymbol{\theta}_t^i$, which are encoded into $\boldsymbol{s}_t^i$, are obtained from the local optimization procedure, i.e., the minimization of the local objective using the data set $\mathcal{D}^i$. Hence, the sensitivity to changing a sample may differ depending on the local optimization algorithm.

### Privacy-preserving encoding methods

Existing federated learning approaches to preserving privacy commonly use either secure encryption or artificial noise obscuring. Each method is characterized by a different tradeoff relevant to the unique federated learning challenges, as discussed in the following:

1) Encryption for federated learning typically relies on additive homomorphic schemes. This approach exploits the fact that the model updates are often aggregated by

## Dithered Quantization for Federated Learning

*Universal quantization* refers to discretization methods that are invariant of the statistical model of the discretized quantity, and it is typically desirable to quantize such that the distortion is not determined by the input. A common strategy to quantize in such a universal manner is via probablistic quantization, i.e., via dithering [22]. Applying scalar dithered quantization, that is, adding noise to each element of the model update, followed by uniform quantization, results in the quantized stochastic gradient descent (QSGD) method proposed in [7]. The application of subtractive dithered quantization, as proposed in [8], can be achieved for both scalar and vector quantizers by requiring the server to share a source of common randomness, e.g., a random seed, with each user.

The universal vector quantization for federated learning (UVeQFed) scheme of [8] consists of the following steps. First, an $L$-dimensional lattice $\mathcal{L}$ is fixed. Upon local encoding, each device normalizes its local model updates (11) by $\zeta$ times its norm, where $\zeta > 0$ is a given scaling factor. The normalization result is divided into $M$ $L$-sized vectors to which the users add a random dither signal randomized in an independent identically distributed (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 server decompresses the model updates by recovering the lattice point and subtracting the dither signal from it, as shown in Figure 4. The fact that the users and the server can generate the same dither signals relies on their ability to share a source of common randomness. When the dither is not subtracted at the server and the lattices are scalar, i.e., $L = 1$, UVeQFed specializes the QSGD [7], which is simpler to implement but results in increased distortion.

Using randomized lattices results in a compression scheme that is simple to implement and invariant of the model updates statistics and that maintains the theoretical performance guarantees of local SGD carried out without any compression. In particular, combining local SGD with UVeQFed achieves the same convergence profile as in

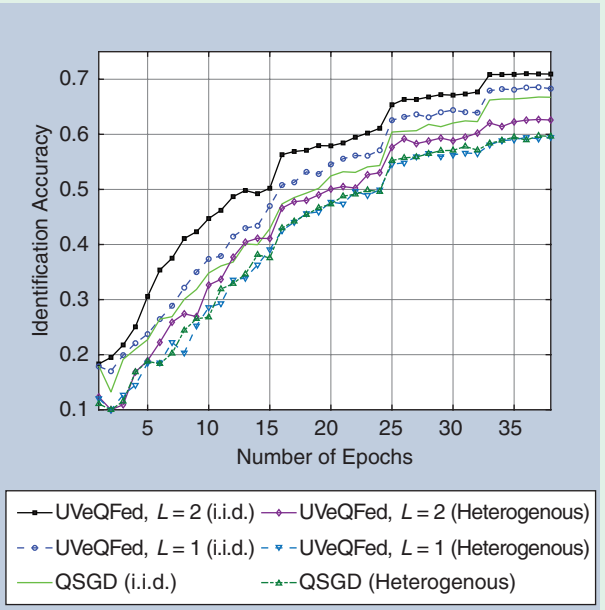



**FIGURE S1.** The accuracy versus the number of epochs of federated averaging combined with universal vector quantization, assigning one bit per model parameter; adapted from [8]. The setup considers the training of a five-layer convolutional neural network using the Canadian Institute for Advanced Research-10 data set with both i.i.d. and heterogeneous data division among $N = 10$ devices. The compared schemes are UVeQFed with 2D lattices $(L = 2)$ and 1D lattices $(L = 1)$ and QSGD [7], which implements the encoding as UVeQFed, with $L = 1$, without subtracting the dither at the server side.

(S1) for the strongly convex, smooth objective with bounded gradients scenario (see "Convergence of Local Stochastic Gradient Descent"). The effect of the compression mechanism results in $\tilde{B}$ replaced with $\tilde{B} + 4M\zeta^2\bar{\sigma}_{\mathcal{L}}^2 E^2 \sum_{i=1}^N p_i^2 \sigma_i^2$, where $\bar{\sigma}_{\mathcal{L}}^2$ is the normalized second-order moment of the lattice $\mathcal{L}$. This yields the same asymptotic convergence behavior of $\mathcal{O}(1/T)$ as local SGD without compression. Furthermore, the combination of subtractive dithering and multivariate lattice quantization results in improved accuracy of the learned model when training deep neural networks with nonconvex loss surfaces, as demonstrated in Figure S1.

averaging to help ensure privacy preservation while maintaining the FedAvg flow [27]. However, the encrypted updates are of a larger size than the plain updates, resulting in an increased communication factor. Additional forms of encryption for federated learning that do not involve homomorphic encryption have been proposed, e.g., [28], though with added communication overhead and possibly limited compliance to the presence of stragglers

## Privacy Boosting by Noising Before Model Aggregation

Consider a federated learning system consisting of one server and $N$ users, where each user holds a data set with size $|\mathcal{D}^i| = n_i$. The server and users are assumed to be honest; however, there are external adversaries targeting users' private information. There are at most $\ell(\ell \le T)$ exposures of uploaded parameters from each user in the uplink, where $T$ is the number of aggregation times. The goal is to implement a federated learning system subject to an $(\epsilon, \delta)$ differential privacy constraint defined as follows: a randomized mechanism $\mathcal{M}: \mathcal{X} \to \mathbb{R}$ satisfies $(\epsilon, \delta)$ differential privacy if for all measurable sets $\mathcal{S} \subseteq \mathbb{R}$ and any two adjacent data sets $\mathcal{D}^i \tilde{\mathcal{D}}^i \in \mathcal{X}$, $\mathbb{P}(\mathcal{M}(\mathcal{D}^i) \in \mathcal{S}) \le e^{\epsilon}\mathbb{P}(\mathcal{M}(\tilde{\mathcal{D}}^i) \in \mathcal{S}) + \delta$.

The noising before model aggregation federated learning (NbAFL) scheme proposed in [29] uses the Gaussian mechanism to guarantee $(\epsilon, \delta)$ differential privacy; i.e., the mapping (8) is in the form of

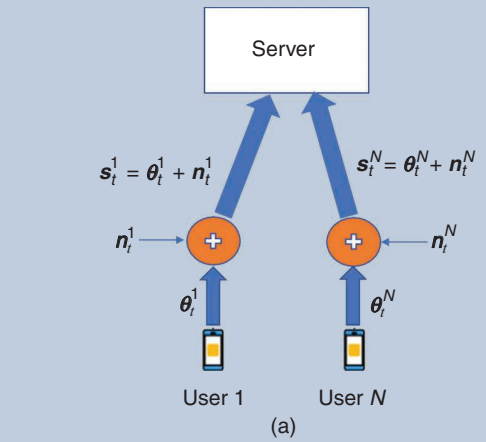
$$\boldsymbol{s}_t^i = \boldsymbol{\theta}_t^i + \boldsymbol{n}_t^i, \tag{S2}$$

where $\boldsymbol{n}_t^i$ is an independent Gaussian noise, whose elements are independent and identically distributed and drawn from $\mathcal{N}(0, \sigma_i^2)$. To achieve the differential privacy constraints using the Gaussian mechanism, it is required to choose $\sigma_i \ge c\Delta s^i/\epsilon$ and the constant $c \ge \sqrt{2\ln(1.25/\delta)}$, where $\epsilon \in (0,1)$ [26] and $\Delta s^i = \max_{\mathcal{D}^i, \tilde{\mathcal{D}}^i} \| s(\mathcal{D}^i) - s(\tilde{\mathcal{D}}^i) \|$ is the local sensitivity.

Combining this with the result from [26] given in the preceding, the sufficient variance noise for the users, $\sigma_i$, is obtained by

$$\sigma_i = \frac{4c\ell C}{\epsilon \cdot \min\{n_i\}}, \forall i, \tag{S3}$$

where $C$ is a clipping threshold for bounding $\boldsymbol{\theta}^i$, satisfying $\|\boldsymbol{\theta}^i\| \le C$, $\forall i$. As shown in [29], the expected gap between the achieved loss function by NbAFL and the minimum loss for a fixed number of users is a decreasing function of $\epsilon$. This illustrates the tradeoff between privacy and accuracy: by increasing $\epsilon$, the degree of privacy is decreased; however, this results in less noise variance, as can be seen from (S3), and hence the performance of the algorithm improves, as detailed in Figure S2(b). Furthermore, (S3) also shows that by increasing the size of the data set $n_i$, one can achieve a predefined level of privacy $\eta$ with less variance of the privacy-preserving noise, thus degrading the accuracy of the learned model less. Finally, since the server averages $\{\boldsymbol{s}_t^i\}_{i=1}^N$ into a global model, the effect of the independent identically distributed privacy-preserving noise on the aggregated model decreases as $N$ grows, and hence the model performance increases accordingly [29].



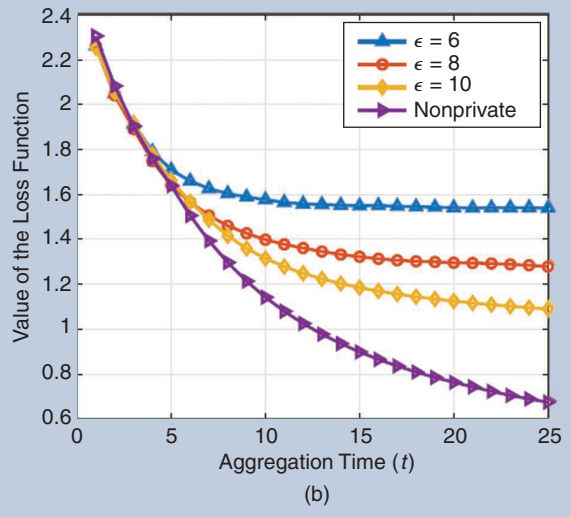




**FIGURE S2.** The NbAFL scheme proposed in [29] for federated learning-based differential privacy with Gaussian artificial noise. The numerical study in (b), adapted from [29] with the authors' permission, considers federated learning on the standard Modified National Institute of Standards and Technology data set for handwritten digits, where the loss function is compared under different protection levels $\epsilon$. (a) Privacy-boosted federated learning via NbAFL. (b) The convergence profile of NbAFL.

and device participation. In general, encryption methods involve additional communication overhead as well as possible secret key sharing between the users [27].

2) Artificial noise implies that each user perturbs its local updates and sends only a randomized version to the server to protect against private information leakage [29]. The common criterion used for characterizing privacy guarantees is differential privacy, being an indicator of the ability to recover a single data sample from the obscured model updates.

Adding artificial noise for differential privacy considerations naturally gives rise to a tradeoff between privacy and convergence performance. A high noise variance makes it more difficult for attackers to recover the information but, at the same time, degrades the convergence of the learned model. Nonetheless, for a given desired privacy level, which dictates the variance of the artificial noise obscuring the model updates, increasing the number of participating users $N$ mitigates the convergence degradation due to privacy preservation, as the effect of the artificial noise vanishes in averaging [29]. Furthermore, it was observed in [29] that for each protection level, there exists a specific, finite setting of the number of local iterations $E$ that optimizes the convergence performance for a given protection level. Thus, the design of privacy-preserving algorithms should be carefully considered to ensure the consistency of the optimization problem.

Two popular mechanisms for achieving differential privacy in federated learning encode the model updates by corrupting them with Laplacian and Gaussian artificial noise signals [29]. Differential privacy can also be achieved with nonartificial noise through the channel noise induced in uplink transmission when using uncoded transmissions, as shown in [30]. An example of a differential privacy scheme is available in "Privacy Boosting by Noising Before Model Aggregation." Finally, it is noted that artificial noise obscuring and encryption can also be combined in a hybrid manner to jointly guarantee a stronger degree of privacy preservation in federated learning, as proposed in [31].

## Uplink transmission

The second stage in the global aggregation step of federated learning involves the transmission of the encoded updated models, produced as the outputs of the local encoding stage, to the server. This operation is essentially a communication procedure, as it requires multiple devices to convey information to a remote server in a reliable manner and with minimal delay. Here, we present the conventional transmission strategy and discuss the challenges that arise from this approach. We then discuss signal processing and communication methods for dealing with similar challenges and elaborate on their adaptation for the federated learning setup.

Federated learning typically involves a large number of edge users, which often reside in various physical locations and utilize different infrastructures for communicating with the server. In particular, some of the users may communicate over the same shared channel, e.g., a set of mobile devices served by the same wireless access point, while some can utilize nonshared wireline links, as in Figure 2. As we discuss in the following, the main challenges associated with the uplink transmission stage arise when the users communicate over a shared medium, e.g., when the users are wireless transceivers served by the same access point. Therefore, to highlight the potential benefits of properly designed uplink transmission schemes, we focus on an extreme scenario in which all the users communicate with the server over the same wireless channel. Nonetheless, the methods detailed next can be adapted and contribute to federated learning networks where distinct subsets of the users share the uplink channel.

### *Conventional transmission*

From an implementation perspective, it is often simpler to separate the communication functionality and the learning application. In this case, the learning procedure is agnostic to how the model updates are communicated, while relying on an existing communication architecture to provide these transmissions. Conventional communication architectures are ignorant of the learning task and enable multiple users to share a channel by mitigating interference via resource allocation. This is achieved by dividing the channel resources into multiple resource blocks and assigning each block to a different user. A common approach is to split the available bandwidth into nonoverlapping frequency bins via orthogonal frequency division multiplexing multiple-access protocols. Such orthogonlization results in each user having a separate channel with the server; i.e., $\mathbb{P}(\boldsymbol{y}_t^1, \ldots, \boldsymbol{y}_t^N | \{\boldsymbol{x}_t^i\}) = \prod \mathbb{P}(\boldsymbol{y}_t^i | \boldsymbol{x}_t^i)$. Then, reliable communication is achieved by having each user employ a channel code with a code rate not larger than the capacity of an individual channel, guaranteeing that the server can accurately recover $\boldsymbol{s}_t^i$ from $\boldsymbol{y}_t^i$ with arbitrarily high probability.

While the conventional strategy allows reliable transmission of the model updates from the users to the server, it can induce notable communication delay. This delay is sometimes more dominant than the duration for each user to carry out its local optimization, thus affecting the convergence of the learned global model. To see this, recall that wireless channels are limited in bandwidth and energy, and thus the achievable rate of each user decays with the number of participating devices, as the channel resources are divided among more of them. Since the number of devices taking part in the federated learning procedure is typically very large, while the model updates are often comprised of a massive number of parameters, the delay of each user can be quite significant. Furthermore, the delays may vary considerably between different users, due to the heterogeneous nature of edge devices, notably affecting the overall delay of each uplink transmission round, being dictated by the user with the longest delay. Since federated learning is composed of multiple rounds, the conventional strategy is likely to result in a lengthy convergence time of the learning procedure, which is translated into degraded accuracy of the models being learned in a given finite time.

### Signal processing perspective

Mapping the encoded model updates $\boldsymbol{s}_t^i$ into a channel input $\boldsymbol{x}_t^i$ consists of two main tasks: the first is the traditional communication operation of formulating a channel input that achieves reliable communication at the maximal rate, given the assigned channel resources via, e.g., modulation and channel coding; the second, which encapsulates much of the potential of minimizing the transmission delay, is a problem of resource allocation [32]. Resource allocation is well studied in the signal processing and communication literature, considering various objectives, including throughput and energy efficiency. However, unlike conventional communication systems, where the receiver is interested in recovering the transmitted messages, in federated learning, the server needs the model updates only as an intermediate step in producing the global model. This gives rise to learning-aware resource allocation and transmission schemes, which are detailed in the following. In particular, we first discuss how the resources of the wireless channels can be divided among the users while accounting for the learning task. Then, we present the notion of over-the-air federated learning, allowing the users to share the same resources, i.e., to communicate in a nonorthogonal fashion, while treating the effect of the channel as a form of functional network over-the-air computation [33].

### Learning-aware resource allocation

The uplink channel is comprised of multiple resource blocks. These blocks represent, e.g., portions of the bandwidth and temporal transmission slots. While dividing these resources equally among all users is likely to limit the convergence rate of the learned global model, one can notably improve the convergence rate by dividing the resources in a manner that accounts for the overall learning procedure.

## Resource allocation in wireless networks

Allocating resources in wireless networks with multiple users is comprised of user selection and resource management. The former deals with the selection of the users to which resources are allocated, i.e., the users that are allowed to transmit. The latter refers to the division of the resources among the selected users. In the conventional signal processing literature on resource allocation, which deals with communication over wireless networks that is not necessarily for training parametric models, user selection and resource management are commonly studied jointly. Common resource allocation strategies are categorized into static predefined allocation and dynamic divisions. Leading design tools include optimization-based strategies, which formulate the resource allocation problem by using a network utility function, and game-theoretic approaches that employ a local utility for each user [32]. These utility measures typically characterize the communication rate, fairness, and energy efficiency. Here, partial participation, i.e., user selection, is obtained as a byproduct of this utility, which may result in some users not employing any channel resources.

## Design considerations

Unlike resource allocation in conventional wireless networks, partial user participation is typically desirable in federated learning regardless of the availability of channel resources. As a result, user selection and resource management are often considered separate tasks, as opposed to the conventional resource allocation literature.

### User selection

The need to limit the number of users that participates in each global aggregation round was already identified in the original federated learning paper, by McMahan et al. [2]. The motivation for user selection stems not only from the communication bottleneck but also from reasons associated with computation and load balancing. It was noted that increasing the number of participating users beyond some point no longer contributes to the learned model [2] and that accuracy similar to that achieved using full device participation can be obtained while having each device participate in a subset of the global aggregation rounds [15]. These observations capture the main distinction between the communication problem of uplink transmission, in which the goal is to convey the messages from all the users to the server, and uplink transmission in federated learning, where messages are useful only if they contribute to the learned global model.

The user selection problem deals with determining the set of participating devices, denoted $\mathcal{G}_t$. Each user that is not part of this set does not participate in the aggregation at iteration $t$, and thus

$$i \notin \mathcal{G}_t \Leftrightarrow \varphi_i(\cdot) \equiv 0. \tag{12}$$

The design of a user selection method must account for the following considerations:

1) Knowledge of $\mathcal{G}_t$ is essential for computing the averaged global model via (6). Consequently, $\mathcal{G}_t$ is typically determined at the server side and is distributed in the downlink transmission, implying that users that are not selected do not need to carry out local training.
2) Since the essence of user selection is to limit the number of participating devices, the process must result in the cardinality of $\mathcal{G}_t$ being bounded; i.e., $|\mathcal{G}_t| < K$ for each $t \in \mathcal{I}_T$ and for some $K \leq N$, preferably $K \ll N$.
3) To allow all devices to participate in the learning procedure, user selection should also involve a scheduling policy such that $\cup_{t \in \mathcal{I}_T} \mathcal{G}_t = \mathcal{N}$, where the equality either holds directly or with high probability.

### Resource management

Once the set of participating users is determined, the channel resources are allocated among the devices. As opposed to user selection, which is often employed regardless of whether the devices share the same channel or not, resource management is relevant only for users that communicate over the same media. In particular, resource management deals with the division of the channel resource blocks, e.g., frequency bins, among the users in $\mathcal{G}_t$. The resource blocks allocated at iteration $t$ to

user $i \in \mathcal{G}_t$ to determine the iteration's transmission mapping $\varphi_i(\cdot)$ via, e.g., its spectral support.

Schemes for managing channel resources should account for the following considerations:

1) Resource reuse, i.e., assigning the same resource block to more than one user, inevitably yields some level of interference. While cross interference can sometimes be exploited, as detailed in the "Over-the-Air Federated Learning" section, it is often preferable to avoid such reuse, and thus the number of resource blocks should not be smaller than $|\mathcal{G}_t|$.
2) The participating users in federated learning are often heterogeneous in their system capabilities and energy, motivating an uneven division of the channel resources.
3) The objective of resource management is to allow rapid and reliable transmission of the messages $\{\boldsymbol{s}_t^i\}_{i\in\mathcal{G}_t}$. Consequently, the measure of interest here is the maximal delay in conveying these fixed-length messages, which is fundamentally different from typical communication measures, such as the achievable sum rate.

### Resource allocation methods

Based on the preceding considerations, one can mitigate the effect of some of the challenges associated with federated learning by properly selecting the users that participate in each round and then dividing the channel resources among them in a learning-aware manner. While the objectives may differ from those for conventional resource allocation, one can still utilize existing methods with proper adaption to account for the learning task. Resource allocation methods for federated learning include the following considerations:

1) User selection can be carried out via deterministic scheduling, random selection, or using an online policy. For instance, a deterministic scheduling policy can assign the most computationally weak devices to participate in the same round, thereby reducing the delay induced by device heterogeneity by decreasing the local processing delay of the remaining rounds.

   In random selection, a fixed number of $K \leq N$ users is randomized such that $|\mathcal{G}_t| = K$ and $\{\mathcal{G}_t\}_{t\in\mathcal{I}_T}$ are mutually i.i.d. A simple strategy is to implement such random selection in a uniform fashion, i.e., having $\mathcal{G}_t$ randomized consistently from all subsets of $\mathcal{N}$ of cardinality $K$ [2], [34]. This simple strategy, which treats all users in a similar manner, can be proved to yield an asymptotic convergence profile similar to that of FedAvg with full device participation [15]. Nonetheless, one can incorporate some knowledge about the characteristics of each user and its contribution to the training procedure by deviating from uniform selection.

   For instance, by modifying the distribution based on which $\mathcal{G}_t$ is randomized from $\mathcal{N}$ to account for the contribution of each user to the learning procedure and its expected delay, one can reduce the convergence delay without affecting accuracy when compared to full device participation [13], [35]. For a detailed example, see "Delay Minimization With Probabilistic User Selection." An alternative nonuniform approach is to use importance sampling to adaptively assign probabilities to users in each round. It was shown in [36] that implementing random device selection via importance sampling based on users' computing, communication, and data resources can notably increase the convergence speed.

   Online scheduling selects the participating users in each round by accounting for the past selected devices and their observed behavior, e.g., delay. Such selection policies can be designed by adopting a multiarmed bandit framework [37]. Here, the users are selected to strike a balance between the exploitation of those that performed well in the past, e.g., those that transmitted with little delay, and the exploration of those that have not sufficiently participated until the current round. For instance, the authors of [37] assigned each user an index consisting of average transmission times based on previous rounds (the exploitation term) and a second term that was inversely proportional to the number of rounds the device transmitted (the exploration term). The latter ensures that users that have transmitted less are given priority to capture potential environment changes that may reduce their transmission delay. In each round, the server selects the set of $K \leq N$ users with the highest indexes.
2) Resource management can be implemented in a centrally controlled manner, optimizing a centralized utility, as in conventional resource allocation. This utility function accounts for measures of the learning task, e.g., the model convergence and training time [14]. Here, the allocation of the resource blocks can be dictated either by the server or the wireless access point via which the sharing users communicate with the server, and it is conveyed to the users during the model distribution step. An example of a resource management scheme that accounts for delay minimization is presented in "Delay Minimization With Probabilistic User Selection." Alternatively, one may allow the users to employ the channel resources in a decentralized, opportunistic manner, harnessing, e.g., dynamic spectrum access techniques [38].

## *Over-the-air federated learning*

An alternative strategy for uplink transmissions in federated learning over shared channels, which can be treated as an extreme case of resource management, is to allow full reuse of all the channel resource blocks. Here, the users simultaneously employ the complete temporal and spectral resources of the uplink channel in a nonorthogonal manner. The rationale is to exploit the inherent aggregation carried out by the shared channel as a form of over-the-air computation [33].

### Over-the-air functional computation

The concept of over-the-air functional computation originated in coding studies for multiple-access channels [39]. It is based on the counterintuitive finding that interference can be

## Delay Minimization With Probabilistic User Selection

Consider a set of users that communicate with the server via an access point over a shared wireless channel comprised of $K$ distinct resource blocks of bandwidth $b$. When the $k$th resource block is allocated solely to user $i \in \mathcal{N}$, the device communicates with the server over a flat fading channel with Gaussian noise and interference, given by

$$y_i = h_i x_i + w_i + v_k. \tag{S4}$$

Here, $x_i$ is the channel input with maximal transmit power $P$; $h_i$ is the channel gain of the $i$th device, satisfying $h_i \propto d_i^{-2}$, where $d_i$ is its distance to the access point; $w_i$ is white Gaussian noise with variance $\sigma_w^2 b$; and $v_k$ is the interference in the $k$th resource block, whose energy is $\sigma_{v,k}^2$. The data rate over the channel (S4) is given by $R_{i,k} = b \log_2(1 + (h_i P/\sigma_{v,k}^2 + \sigma_w^2 b))$. The learning-aware scheme proposed in [13] allocates the resources for such uplink transmissions in two stages.

**Probabilistic user selection**

First, $K$ users are randomly selected to participate in each round. The randomization accounts for both the contribution of each user via the norm of its model updates as well as its transmission delay, making devices with stronger links more likely to participate. Letting $\alpha_t \in [0,1]$ be a parameter balancing these contributions, the participation probability of user $i$ at round $t$ is computed as

$$\rho_t^i = \alpha_t \frac{\left\| \boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i \right\|}{\sum_{i=1}^{N} \left\| \boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i \right\|} + (1-\alpha_t) \frac{\max_j d_j - d_i}{N \max_j d_j - \sum_{i=1}^{N} d_i}. \tag{S5}$$

The probability vector $[\rho_t^1, \ldots \rho_t^N]$ is used to randomize the $K$ participating users $\mathcal{G}_t$.

**Delay-minimizing resource division**

Next, the resource blocks are divided among the users in $\mathcal{G}_t$ by minimizing the transmission delay. By letting $\beta_t^i$ be the bits used for representing the encoded model update $\boldsymbol{s}_t^i$, the delay of user $i \in \mathcal{G}_t$ when assigned the $k$th block is given by $\beta_t^i / R_{i,k}$. Consequently, the resource blocks are allocated to minimize the maximal delay via the following constrained optimization problem:

$$\min_{\{\chi_{t,k}^i\}_{k=1}^K \in \{0,1\}^K} \max_{i \in \mathcal{G}_t} \frac{\beta_t^i}{\sum_{k=1}^{K} \chi_{t,k}^i R_{i,k}}; \quad \text{subject to} \sum_{i \in \mathcal{G}_t} \chi_{t,k}^i = \sum_{k=1}^{K} \chi_{t,k}^i = 1. \tag{S6}$$

The constraint in (S6) guarantees that each resource block is assigned to a single distinct user. As shown in [13], the optimization in (S6) can be recast as an integer linear programming problem, facilitating its solution numerically. This combination of contribution- and delay-aware user selection along with delay-minimizing resource allocation enables the learning of accurate models with notably less delay compared to uniform random user selection, as illustrated in Figure S3.

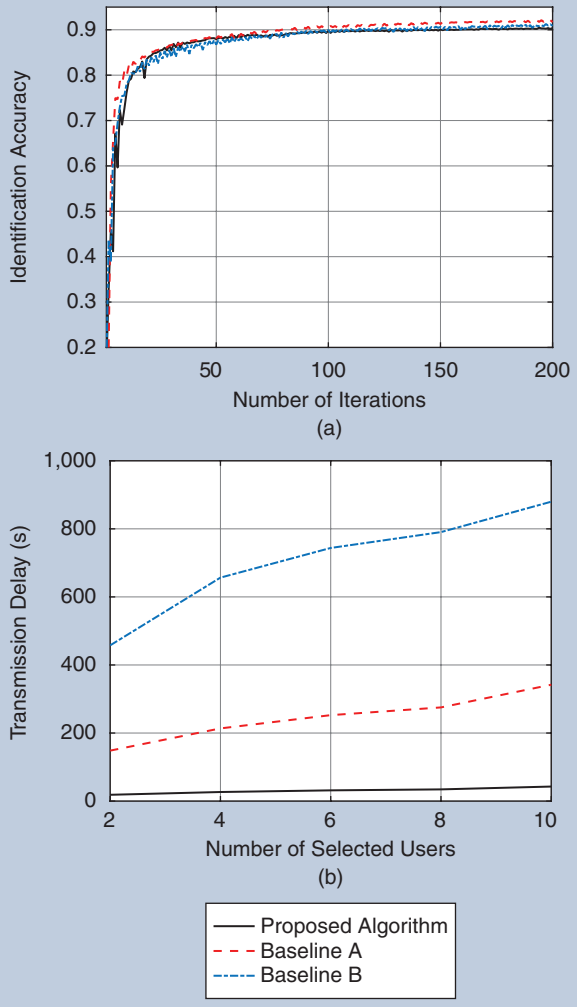



**FIGURE S3.** Federated learning for handwritten digit recognition from the Modified National Institute of Standards and Technology data set distributed among $N = 15$ devices; adapted from [13]. Here, the trained model includes a fully connected neural network with a single hidden layer of 50 neurons. *Proposed Algorithm* and *Baseline A* refer to resource allocation based on delay minimization, while the proposed algorithm is combined with model compression based on [8]; *Baseline B* refers to random resource allocation. (a) The accuracy versus the number of iterations. (b) The convergence time versus the number of selected devices.

harnessed to help computing when there is a one-to-one map between a desired computation and a linear function of the transmissions. Since the emergence of over-the-air functional computation in multiuser communications, the approach has been traditionally studied and developed for statistical inference in wireless sensor networks.

Over-the-air computation relies on having the users share the same channel, as illustrated in Figure 5(a). The main advantage of this approach is that it allows each transmitter to utilize the complete available channel resources, e.g., bandwidth, regardless of the number of users [33]. For the interference result in some desired linear function over a continuous space, such as the averaging of real-valued vectors, the transmitters should employ analog signaling. In such cases, the channel input is a continuous function of the model updates; e.g.,

$$\boldsymbol{x}_t^i = \varphi_i(\boldsymbol{s}_t^i) = \sqrt{\alpha_t^i}\,\boldsymbol{s}_t^i, \tag{13}$$

for some power-scaling parameter $\alpha_t^i > 0$. Using (13), the channel output represents some form of weighted averaging of the transmitted $\{\boldsymbol{s}_t^i\}$. Consequently, transmission is carried out without converting $\{\boldsymbol{s}_t^i\}$, e.g., the model updates, into discrete coded symbols that should be decoded at the receiver side, as is commonly the case in digital communication.

## Design considerations

Over-the-air computations can notably increase the throughput and reduce the transmission delay in federated learning over wireless channels when compared with orthogonal division of the channel resources. However, analog signaling induces challenges to the communication methods. For instance, it requires accurate synchronization among all users. Furthermore, for the users and the server to guarantee that the combining carried out by the shared channel can be transformed into a desired aggregation mapping, knowledge of the channel input–output relationship is typically

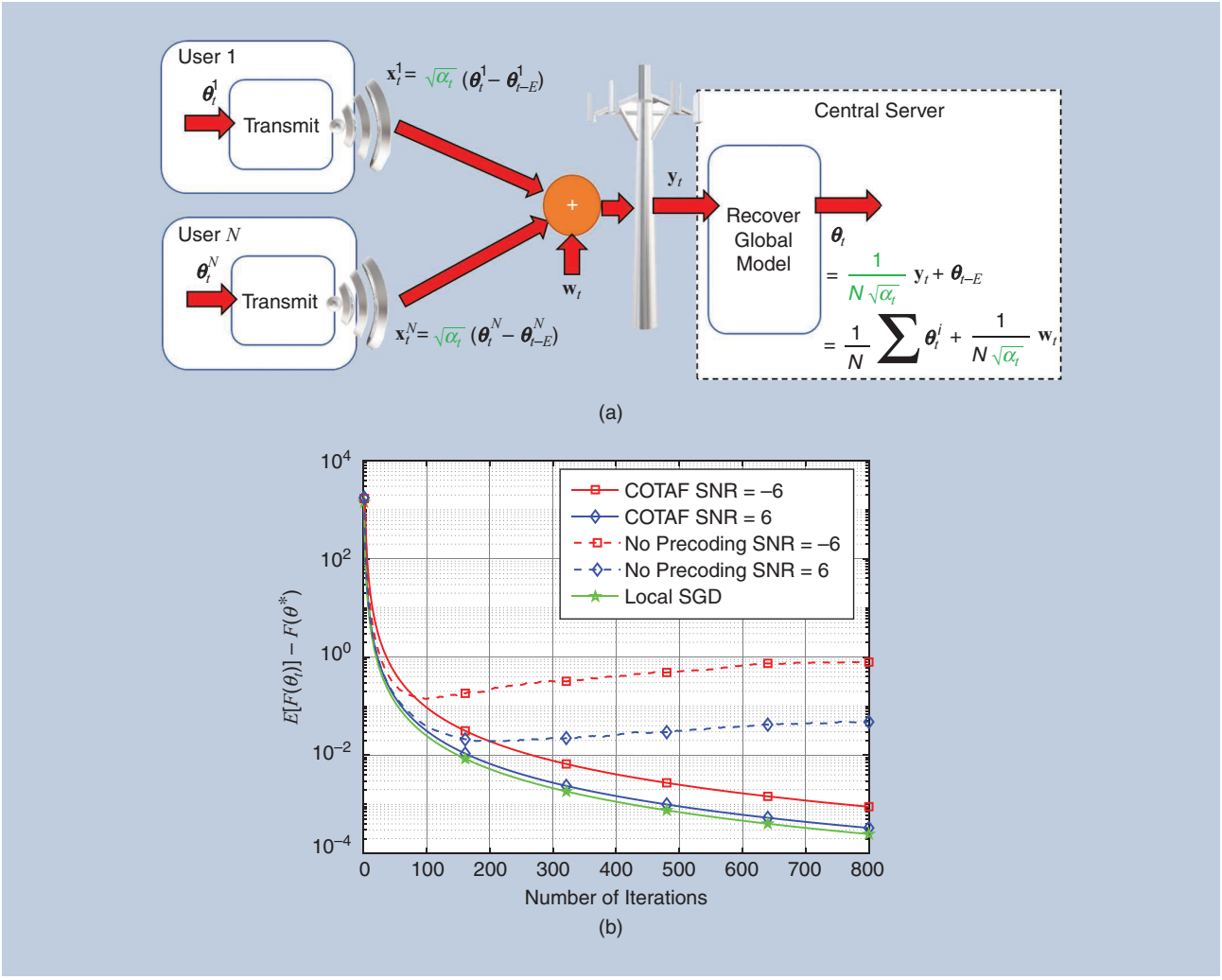



**FIGURE 5.** Over-the-air federated learning. The numerical study in (b) considers the federated learning of a linear predictor using the Million Song data set, where the convergent over-the-air federated learning (COTAF) scheme of [43] is compared to noise-free, local SGD over orthogonal channels as well as over-the-air transmission with no time-varying precoding. (a) Over-the-air federated learning with time-varying precoding (in green). (b) The COTAF convergence profile. SNR: signal-to-noise ratio.

required. Consequently, the design of over-the-air computation methods for federated learning must account for the following considerations:

1) The desired global combining mapping should be one that can be obtained via over-the-air computations. In particular, for linear uplink channel models where the contribution of the interference is additive, the desired aggregation mapping should be a nomographic function. Such multivariate functions can be represented as a univariate functional of a combination of their variables, as is the case for FedAvg aggregation (6). This limits the combination of over-the-air computation with nonnomographic aggregation rules, such as the trimmed mean (see the "Security-Enhanced Federated Combining" section).
2) The transmission mapping $\varphi_i(\cdot)$ and the postprocessing carried out at the server side should mitigate the effect of the channel noise on the learning algorithm. This is necessary to allow the training procedure to converge despite the presence of noise in the model updates.
3) Dedicated mechanisms should be introduced to achieve synchronized transmissions.

The second consideration should be carefully examined for each model since the sensitivity of optimization algorithms is highly dependent on their objective. For instance, SGD-based optimization over convex objectives is highly affected by noisy observations, where convergence can be guaranteed only to some environment of the optimal solution. However, for nonconvex objectives, SGD benefits from a minor amount of noise, which contributes to its convergence by providing means to avoid local minima [40]. To deal with the third requirement, one can incorporate mechanisms utilized in digital communication protocols, which typically employ beacon transmissions and slotted superframes to achieve synchronization. An alternative approach to achieve synchronization is to utilize dedicated schemes designed for over-the-air computations. These include the analog modulation method proposed in [41] and the broadcasting of a shared cloak during the model distribution step, as proposed in [42].

### Over-the-air federated learning methods

The fact that the model updates in federated learning are typically transmitted to be averaged makes over-the-air computation an attractive technique for increasing the throughput when operating over a shared channel. The straightforward application of over-the-air federated learning, in which the transmitters send their stochastic gradients (i.e., local SGD, with $E = 1$) precoded to meet a given power constraint, was shown to result in little convergence delay when learning is carried out over identical unfaded channels with homogeneous data sets [24]. Nonetheless, in more involved setups of federated learning over wireless channels, which include fading, heterogeneous data, and multiple local SGD steps, one can notably improve the accuracy of the learned model and the convergence delay by utilizing dedicated precoding mechanisms as well as integrating methods from the areas of compressed sensing and multiuser multiple-input, multiple-output (MIMO) communication.

The following considerations apply:

1) Dedicated precoding can guarantee convergence of over-the-air local SGD carried out over heterogeneous data sets and with multiple local iterations; i.e., $E > 1$. This is achieved by setting the precoder to account for the fact that the difference in each set of E local iterations is expected to gradually decrease through time [43]. Building on this insight, the model updates are scaled by their maximal expected norm along with a corresponding aggregation mapping at the server side, which jointly results in an equivalent model, where the effect of the noise induced by the channel is mitigated over time. An example of an over-the-air federated learning method that enables high-throughput signaling with guaranteed convergence by the gradual mitigation of the effect of the channel noise is provided in "Time-Varying Precoding for Over-the-Air Federated Learning."

   Additionally, one can precode to mitigate the effect of channel fading in the learned model. For instance, when channel state information is available to the users, they can perform channel inversion precoding, i.e., scaling the model updates by the channel coefficient [10]. In practice, some channels are likely to exhibit deep fading, rendering channel inversion inefficient under a power constraint, and a truncated version of channel inversion is preferable [43]. Here, only users whose channel attenuation is below a predefined threshold transmit, resulting in an additional form of user selection based on channel quality.
2) Compressed sensing tools facilitate exploiting the fact that the model updates typically approach being sparse as the training algorithm progresses [24]. Consequently, when the model updates have a similar sparsity pattern, e.g., when training using homogeneous data sets, the recovery of the aggregated model can be viewed as the reconstruction of a sparse vector. This may be exploited by applying random compressive linear precoding, which can be employed in fading channels without requiring the users to have channel state information, as the receiver recovers the aggregated model via compressed sensing mechanisms [11].
3) Multiuser MIMO methods, which are applicable when the receiver has multiple antennas, can further facilitate high-throughput, over-the-air federated learning by exploiting the spatial diversity of such channels. This can be realized by introducing learning-aware beamforming techniques [44], possibly combined with user selection, guaranteeing that in each round, users sharing a similar relative angle to the receiver take part in global aggregation. Furthermore, when the receiver has a large number of antennas, one can mitigate the need for accurate channel knowledge in over-the-air federated learning by utilizing massive MIMO schemes [45].

## Global combining

The third and last stage in the global aggregation step is the processing and combining of the received signal $\boldsymbol{y}_t$ at the server into the global model $\boldsymbol{\theta}_t$. The goal is to produce an accurate global

model, where accuracy is typically characterized by the global objective (3). We next describe conventional combining strategies. We then discuss how this operation is viewed from a signal processing perspective and provide examples of methods for smart combining in light of the mentioned challenges.

### Conventional combining

FedAvg implements global combining by setting the global model to be a weighted average of the local updates. It is typically assumed that the model updates can be accurately recovered from the channel output $\boldsymbol{y}_t$ such that the server has direct access to the model updates of the users participating in the current round; i.e., $\{\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i\}_{i\in\mathcal{G}_t}$. The global model is then obtained by

$$\boldsymbol{\theta}_t = \Phi(\boldsymbol{y}_t) = \frac{N}{|\mathcal{G}_t|}\sum_{i\in\mathcal{G}_t} p_i(\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i + \boldsymbol{\theta}_{t-E}), \tag{14}$$

where $\mathcal{G}_t$ is the set of participating devices and $\boldsymbol{\theta}_{t-E}$ is the global model broadcast at the previous distribution step.

The weights $\{p_i\}$ in (14) are the same as those in the global objective (4) and typically determined in accordance with the different data set sizes of the users. Nonetheless, these weights often do not account for other differences among the users, such as the heterogeneous nature of the local data distributions, and the fact that each user's instantaneous updates are expected to have a different contribution to the learned model. Intrinsically, the centralized model combining in (14) minimizes the loss of the resulting global model with respect to data drawn from a distribution that equals the weighted average distribution of the local data sets [46].

To formulate this mathematically, let $\mathcal{Q}_i$ be the distribution associated with the data set at user $i$. The model that is optimized based on $F(\cdot)$ minimizes the loss measure taken with respect to data from the averaged distribution $\tilde{\mathcal{Q}} \triangleq \Sigma_{i=1}^N p_i\mathcal{Q}_i$ [47]. This implies that ensuring accuracy in the sense of minimizing the global loss may result in a combined model that is unable to accurately infer samples drawn from the distributions under which it was trained. In addition, conventional strategies assume that the information sent by the users is reliable, ignoring possible adversaries that can harm the results. Straightforward weighted aggregation of the devices' updates can be arbitrarily skewed by a single Byzantine–faulty user via, e.g., poisoning attacks [48].

### Signal processing perspective

The channel output $\boldsymbol{y}_t$ is statistically related to the model updates $\{\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i\}_{i\in\mathcal{G}_t}$. As discussed in the previous sections, this relationship possibly includes noise and attenuation due the transmission procedure as well as distortion due to lossy compression and/or privacy amplification. Consequently, mapping $\boldsymbol{y}_t$ into a global model $\boldsymbol{\theta}_t$, which yields an accurate inference rule in the sense of some risk function, is inherently a statistical estimation setup.

## Time-Varying Precoding for Over-the-Air Federated Learning

Consider an uplink wireless channel shared by $N$ users. When the users share the complete channel resources, i.e., communicate in a nonorthogonal fashion, the channel output at round $t$ is related to the channel inputs $\{\boldsymbol{x}_t^i\}$ via

$$\boldsymbol{y}_t = \sum_{i=1}^N \boldsymbol{x}_t^i + \boldsymbol{w}_t. \tag{S7}$$

Here, $\boldsymbol{w}_t$ is an additive Gaussian noise vector whose entries have zero mean and variance $\sigma_w^2$, $\boldsymbol{x}_t^i$ is subject to an energy constraint $P$, and all vectors are comprised of $d$ elements, i.e., the same as the number of model parameters.

The inherent averaging carried out by the shared channel (S7) can be exploited to enable over-the-air federated learning with local stochastic gradient descent optimization, as proposed in, e.g., [24] and [43]. In this case, the channel input is set to a precoded version of the model updates, given by $\boldsymbol{x}_t^i = \sqrt{\alpha_t}(\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i)$, where $\alpha_t$ is a precoding parameter, which can change between iterations. Under such a setting, the server computes the updated global model via

$$\boldsymbol{\theta}_t = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{y}_t + \boldsymbol{\theta}_{t-E} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{\theta}_t^i + \frac{1}{N\sqrt{\alpha_t}}\boldsymbol{w}_t. \tag{S8}$$

This transmission scheme is presented in Figure 5(a). The authors of [43] proposed to set the precoder to $\alpha_t = P/\max_i \mathbb{E}\{\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i\|^2\}$. This precoding term accounts for the fact that the magnitude of the model updates is expected to decrease as the training process progresses, and thus the setting of $\alpha_t$ results in the global model $\boldsymbol{\theta}_t$ in (S8) being given by $1/N\,\Sigma_{i=1}^N \boldsymbol{\theta}_t^i$, i.e., the desired federated averaging term (6) for $p_i = 1/N$ and $\mathcal{G}_t = \mathcal{N}$, corrupted by an additive noise term whose contribution gradually decays with the iteration $t$.

The time-varying precoding scheme of [43] is referred to as *convergent over-the-air federated learning (COTAF)*. This is because for the strongly convex, smooth objectives with bounded gradients (see "Convergence of Local Stochastic Gradient Descent"), COTAF results in the same convergence profile as in (S1), with $\tilde{B}$ replaced by $\tilde{B} + 4dE^2G^2\sigma_w^2/PN^2$. This implies that the asymptotic convergence behavior of $\mathcal{O}(1/T)$ is the same as that achieved in orthogonal transmissions over noise-free channels, as shown in Figure 5(b), while mitigating the harmful effect of the channel noise and communicating at a higher throughput due the full reuse of the channel resources.

Nonetheless, the application of statistical estimation techniques to global combining is challenging due to some of the properties of federated learning setups. First, while estimation methods often rely on prior knowledge of the joint distribution of the observations and the desired quantity, one rarely has access to a reliable characterization of the statistical relationship between the observations and the desired global model. In fact, it is quite unlikely to assume that the server knows the distribution of the model updates. Furthermore, the global objective in (3) relies on data and is computed using the data sets available at the users, implying that the server cannot even evaluate the global objective for a given model, let alone optimize its combining rule based on it. Moreover, as discussed earlier, even when the server can characterize a combining rule that yields a model that minimizes global objective (3), the resulting representation may be inaccurate for a large family of relevant distributions. Finally, the server should also be able to cope with unreliable users, a requirement that is not typically encountered in statistical estimation.

This motivates the design of dedicated global methods oriented to the federated learning framework. In the following, we first discuss schemes that aim at forming accurate global models from the channel outputs in a nonsecure manner, i.e., without having to handle unreliable and malicious users. Then, we review methods for adapting combining methods to be Byzantine robust, i.e., to defend against malicious users.

## *Federated combining*

The purpose of the combining stage is to integrate the individual users' updated models into one rich global representation, thus gradually refining the inference rule as training progresses. This procedure handles the channel effects, such as the introduction of noise and fading, and aims at having the updated global model yield an accurate inference rule. The direct formulation of this problem uses the global objective (3), with the goal of setting $\boldsymbol{\theta}_t = \Phi(\boldsymbol{y}_t)$ to minimize $F(\Phi(\boldsymbol{y}_t))$. As discussed, when the local data sets are heterogeneous, the model that minimizes $F(\cdot)$ may perform poorly on data drawn from some (or even all) of the distributions used in its training. The effect of statistical heterogeneity can be mitigated by modifying the optimization procedure, effectively resulting in local models corresponding to a distribution other than the local data set distribution $\mathcal{Q}_i$. This can be achieved by, e.g., combining additive local and global correction terms to the local optimization procedure; see, e.g., [3, Sec. 3.2]. Nonetheless, even when the local models are trained using conventional optimizers, which is often the case in federated learning, the harmful effects of statistical heterogeneity can be reduced by combining the model updates, as we show in the following.

### Design considerations

The design of federated combining methods should account for the following considerations:

1) Computing the global loss requires access to the local data sets. In addition, the model updates themselves are determined by the data sets. Since these sets are not available to the server, any optimization carried out using either the global loss measures or a statistical characterization of the model updates inevitably requires additional information exchanges between the users and the server. This should be carried out in a manner that does not induce a notable communication burden and that does not leak information about the individual data samples. Alternatively, one can utilize server-side data, when such is available, for optimizing the combining rule.
2) While edge devices may vary considerably, it is reasonable to assume that those with similar technology and/or geographic locations observe samples of a similar distribution. Tackling heterogeneity in aggregation is notably facilitated if the users can be clustered into groups with similar data distributions, as illustrated in Figure 6(a). For instance, such knowledge can be used to determine the combining rule by treating model updates from various clusters differently. Several methods have recently been proposed to enable clustered federated learning; see, e.g., [49] and [51].

### Federated combining methods

Broadly speaking, there are two main strategies to aggregate the channel outputs into a global model yielding an accurate inference rule. The first is to combine the model updates into a single model $\boldsymbol{\theta}_t$ while using a mapping $\Phi(\cdot)$ that compensates for channel effects, distortion, and/or bias due to heterogeneity. The second strategy aggregates the model updates into a set of global models, i.e., a mixture of models (also referred to as *model interpolation* [52]). During inference, the mappings of these models, rather than their parameters, are combined, as shown in Figure 6(a) and reviewed in the following:

1) *Single combined model*: As discussed in the previous section, the conventional FedAvg combining rule reduces some of the harmful effects of the distortion and noise induced in the encoding procedure and the communication channel, respectively. However, one can further mitigate these effects by properly accounting for their presence. For instance, the authors of [54] proposed to compensate for the distortion induced by coarse quantization by designing the combining rule to approach the minimal mean-square-error estimate of some desirable parameters $\boldsymbol{\theta}_t^{\text{opt}}$; i.e., $\Phi(\boldsymbol{y}_t) \approx \mathbb{E}\left[\left[\boldsymbol{\theta}_t^{\text{opt}}\right] \middle| \boldsymbol{y}_t\right]$. In particular, $\boldsymbol{\theta}_t^{\text{opt}}$ was set to the FedAvg combining rule in [54], while the conditional expectation was computed based on knowledge of the communication channel and the quantization scheme as well as additional information conveyed from the users regarding the statistical moments of the model updates.

   Deviating from the combining rule in (14) can help not only in tackling distortion and noise induced in the encoding and transmission processes but also in mitigating the harmful effects of statistical heterogeneity. This can be achieved via weighted averaging of the weights, as in conventional FedAvg, i.e., $\boldsymbol{\theta}_t = \sum_{i \in \mathcal{G}_t} p_i \boldsymbol{\theta}_t^i$ (for full user participation $|\mathcal{G}_t| = N$), while optimizing the weights $\{p_i\}$. For instance, the agnostic federated learning scheme of [46] learns this combination in a manner that minimizes

the maximal loss over a set of weighting coefficients $\mathcal{P}$; i.e., the loss measure becomes $\max_{\tilde{p}_i \in \mathcal{P}} \Sigma_i \tilde{p}_i f(\boldsymbol{\theta}, \mathcal{D}^i)$. The main drawback in learning the combining weights $\{p_i\}$ along with the model parameters $\boldsymbol{\theta}$ stems from the fact that it involves an increased amount of communication between users and the server.

2) *Mixture of models*: An alternative approach to deal with the harmful effects of statistical heterogeneity in aggregation is to combine the inference rules rather than the model parameters. Here, letting $q_{\boldsymbol{\theta}}(\cdot)$ denote the inference rule parameterized by $\boldsymbol{\theta}$, the global inference rule is given by a combination of $\{q_{\boldsymbol{\theta}^i}(\cdot)\}$. Using a combination of inference rules instead of a single aggregated model facilitates utilizing tools from multisource adaptation, ensemble learning, and the mixture of experts to determine how to combine the models. The main challenge associated with this strategy is that a separate model has to be maintained and trained for each user/cluster of users. This increases the volume of the resulting global model, which is comprised of an ensemble of individual models rather than a single aggregated one, and it implies that the model distributing step over the downlink should be carried out in a multicast fashion, as different models are distributed to various users rather than via conventional broadcasting. An example of a mechanism for combining the inference rules rather than the model parameters is detailed in "Heterogeneous Federated Learning via the Mixture of Models."

Aggregating the parameters into a single model allows all the users to train the same representation. Combining the inference rules is oriented toward scenarios in which inference is carried out at the server side, which has access to all the models. Nonetheless, one can still combine diverse inference rules on the edge via collaboration, as proposed in [53], at the cost of additional communication during inference.

## Security-enhanced federated combining

The fact that many different users take part in federated learning gives rise to security issues that are not commonly encountered in conventional centralized learning. In particular, model updates sent by some users can be unreliable and significantly deviate from their normal values, either unintentionally or maliciously. This, in turn, induces bias on the learned model and may severely degrade the convergence performance. The need to be able to cope with unreliable users falls under

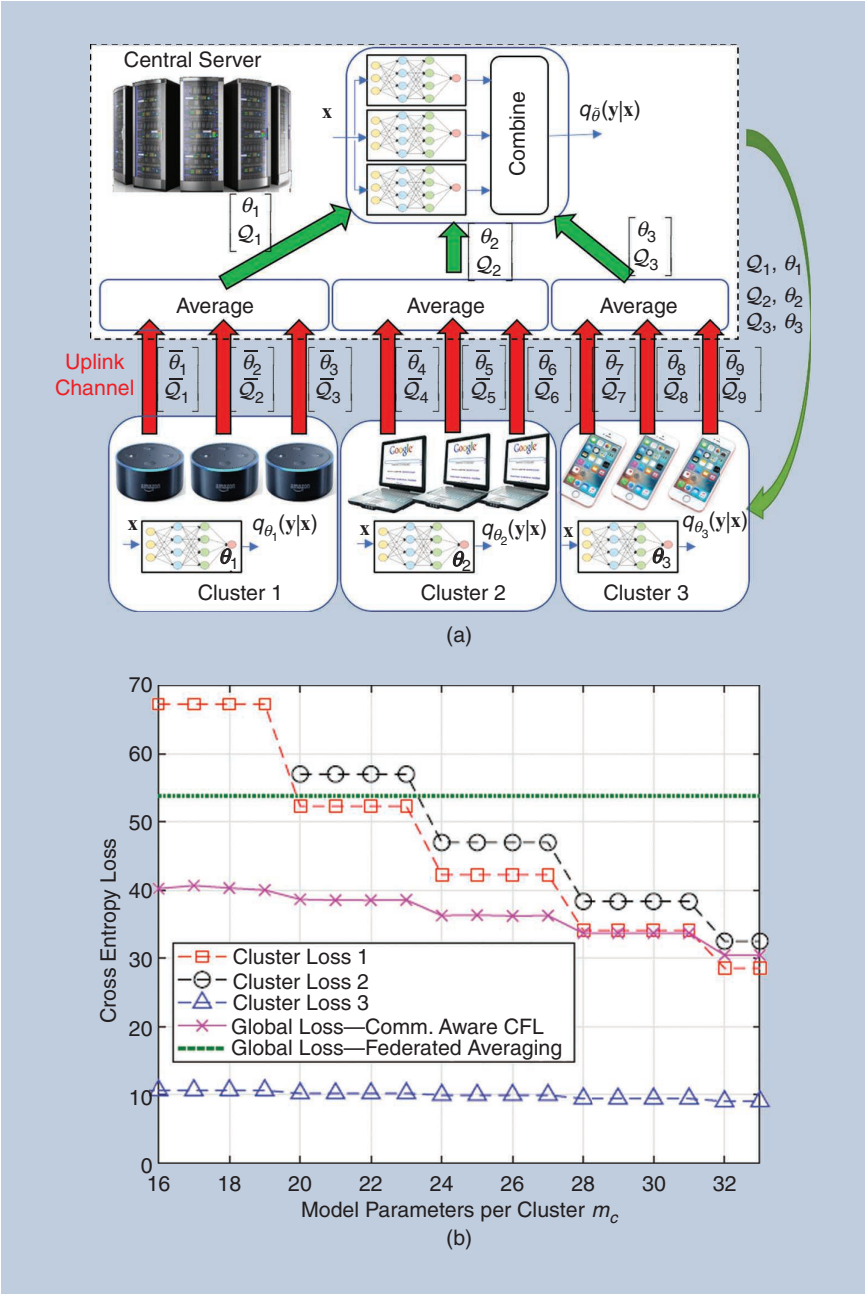



**FIGURE 6.** Clustered federated learning with combined inference rules. The numerical study in (b) considers the training of a linear regression model over data distributed among $N = 9$ users divided into $C = 3$ clusters. In each cluster, the input is randomized in an i.i.d. fashion from a uniform mixture of three identity covariance Gaussian functions. The results compare the accuracy of the combined global model to conventional federated learning based on FedAvg combining when both are tested using data taken from a mixture of the cluster distributions as well that of the individual cluster models tested on data from their local distribution. (a) Clustered federated learning. (b) The loss versus the model parameters [55]. CFL: communication-aware clustered federated learning.

the framework of distributed robust learning [56], also referred to as *Byzantine fault-tolerant learning* or *Byzantine–robust learning*.

Distributed robust learning is concerned with samples containing a mixture of authentic samples and outliers held by a central server. The authentic samples are generated according to an underlying model (i.e., the ground truth), and the outliers may be arbitrarily corrupted or even maliciously chosen. Such robust aggregation methods are typically nonaffine and designed not to be affected by the values of the extreme (and possibly unreliable) samples. A common way to reduce the effect of the malicious outliers on the learned model is to use the geometric median in the aggregation procedure instead of the simple averaging. The geometric median is a generalization of the median in one dimension to multiple dimensions. It aggregates a collection of independent estimates into a single approximation with significantly stronger concentration properties even in the presence of a constant fraction of outliers in the collection.

### Design considerations

For a federated learning aggregation method to be Byzantine fault tolerant, it must not be affected by abnormal model updates, which correspond to unreliable users. A key challenge which arises in Byzantine–robust federated combining stems from the fact that the aggregation technique should not only mitigate the effect of abnormal updates values but do so in a way that does not significantly impair the optimization performance and without inducing additional significant communication between the users and the server. Thus, an important question in this setting concerns what levels of accuracy one should expect when training a model in a federated manner while being Byzantine robust and how to design algorithms that improve accuracy.

In particular, the design of Byzantine–robust federated combining methods must account for the following considerations:

1) The heterogeneous nature of the users must be accounted for when boosting Byzantine fault tolerance. Specifically, not only are the users outside the control of the server but data may not be homogeneous among devices, and different users possess various amounts of data. Thus, a relatively large level of diversity, which is typically used in Byzantine–robust distributed optimization as a measure of abnormality, is expected in federated learning even when all the users are reliable.
2) Unreliable users may not only provide corrupted model updates but also affect the aggregation rule. For instance, conventional FedAvg weights the updates based on the number of samples each user has since $p_i$ in (6) is usually set to $p_i = n_i / \Sigma_j n_j$. Unreliable users may thus report a false data set size to modify the aggregation rule. This ability of Byzantine–faulty entities to corrupt not only their own contribution but also how it is processed by the server is a unique challenge arising in federated learning.
3) In general, Byzantine–robust combining methods require knowledge of the maximal fraction of unreliable users. Nonetheless, some methods are invariant to this requirement, as discussed in the following.

## Heterogeneous Federated Learning via the Mixture of Models

Consider a federated learning system with $N$ users, where the data at the $i$th device follows a marginal distribution, denoted $\mathcal{Q}_i(\cdot)$. Each user employs its local data set $\mathcal{D}^i$ to train a local model $\boldsymbol{\theta}_i^i$ that dictates an inference rule $q_{\boldsymbol{\theta}_i^i}(\boldsymbol{a})$ when applied to sample $\boldsymbol{a}$. The users are divided into $C \leq N$ clusters (preferably $C \ll N$), where each cluster is characterized by a single input distribution measure $\mathcal{Q}_i(\cdot)$.

In combining via the mixture of models, the server orchestrates the training of a separate model for each cluster via conventional federating averaging (FedAvg). In such cases, $C$ separate global models are simultaneously trained in a federated manner [51] using (approximately) homogeneous data sets. By letting $\tilde{\boldsymbol{\theta}}_i^c$ be the global model parameters of the $c$th cluster, the global inference rule is given by $q_{\boldsymbol{\theta}}(\boldsymbol{a}) = \Sigma_{c=1}^{C} \alpha_c q_{\tilde{\boldsymbol{\theta}}_c}(\boldsymbol{a})$, where $\{\alpha_c\}$ are the averaging coefficients.

The authors of [55] proposed to use averaging coefficients, which are parametric estimates of the cluster marginals; i.e., each $\alpha_c$ is a function of the input $\boldsymbol{a}$ such that $\alpha_c(\boldsymbol{a}) \approx \mathcal{Q}_c(\boldsymbol{a}) / \Sigma_{k=1}^{C} \mathcal{Q}_k(\boldsymbol{a})$. This implies that an additional model for estimating the marginal distribution is trained for each cluster, at the cost of additional communication overhead. An illustration of this procedure with $N = 9$ users and $C = 3$ clusters is given in Figure 6(a).

Such aggregation can be shown to yield reliable inference when applied to samples from different distributions that are within some proximity to the cluster marginals $\{\mathcal{Q}_c\}$. In particular, if the loss measure of the model of cluster $c$ is upper bounded by $\epsilon_c$, then for samples drawn from any distribution that is $\rho$-norm bounded (see [47]) by $\{\mathcal{Q}_c\}$, the expected loss can be upper bounded by $(1 + \delta / 1 - \delta)\, \rho\, \Sigma_{c=1}^{C} \epsilon_c$, where $\delta > 0$ is a bound on the difference between the estimated cluster marginals and the true ones (see [55] for the exact statement of the difference). This indicates that one can achieve an arbitrarily small global loss for the individual distributions observed at each cluster in "test" time. Another insight arising from the result is that the individual cluster loss is directly related to the richness of its model. This is illustrated in Figure 6(b), which also compares combining based on FedAvg.

### Robust combining methods

FedAvg combining is ignorant of the value of the model updates and is thus sensitive to corrupted models. Therefore, Byzantine–robust combining methods must account for the specific values of the updates and provide means for identifying which of the updates are treated as abnormal. The leading methods to realize robust aggregation are based on replacing the conventional averaging with one of the following nonaffine computations:

1) Median aggregation is highly robust to abnormal updates. For instance, the 1D median operation has the property that if more than half the sample points lie in some range $[-r, r]$ for a given $r > 0$, then the median must be in $[-r, r]$. Likewise, in multiple dimensions, the geometric median has a similar robust property [56], [57]. The method of geometric median aggregation can be further generalized to marginal median aggregation and to "mean-around-median" aggregation [58]. When used for aggregating the model updates, median-based aggregation guarantees that the difference in the global model is in the proximity of at least half the considered updates, which yields resiliency against abnormal updates. Such robust aggregation does not require knowledge of the fraction of unreliable users.
2) Krum aggregation bears some similarity to median aggregation in the sense that it selects a single model update to use for updating the global model. Here, the model update is selected as the one that is the closest to a given set of neighboring model updates.
3) Truncation mapping first discards a subset of the model updates, which are treated as abnormal, and then aggregates the remaining updates via FedAvg [57].

The preceding methods are designed to constitute a robust alternative to mere averaging. They can also be modified to replace weighted averaging, e.g., by applying imbalanced data sets to the weighted model updates scaled by their proportional coefficient $N \cdot p_i$. Furthermore, the challenge with dealing with diversity due to heterogeneity in a Byzantine–robust manner can be tackled by dividing the users into clusters and aggregating in a robust manner in each cluster separately [51]. An example of Byzantine–robust federated learning methods using the median and truncation combining strategies is given in "Byzantine–Robust Federated Learning."

## Future research directions

We conclude the article with a discussion of some of the future research directions that can further strengthen the role of signal processing and communications in the emerging federated learning paradigm. We follow the division of the global aggregation step used in the article, discussing future research directions related to the users' local updates encoding, after which we elaborate on directions focusing on the transmission of the encoded model updates and their combining on the server side. Finally, we discuss additional future directions that go beyond this division.

### *Local updates encoding*

Processing the updated models produced by the local optimization procedure enables us to tackle some of the core challenges of federated learning. As discussed in the "Local Updates Processing and Encoding" section, by encoding the model updates prior to their transmission, one can notably relieve the communication burden via, e.g., quantization and compression as well as boost privacy preservation using encryption and differential privacy techniques.

The potential of learning-aware local encoding in facilitating federated learning at a large scale gives rise to a multitude of possible research directions. Most studies of uplink compression to date focus on a single user, assuming that all devices in the network employ the same encoding mechanism. Nonetheless, the heterogeneous nature of federated learning motivates treating the need to compress the model updates from a perspective of the overall distributed system. For instance, one would expect users of various technologies to harness a different quantization resolution, as for each device, the resolution is expected to be translated into a communication delay in a different manner. Furthermore, it may be preferable to change the compression mapping as the learning procedure progresses, using different resolutions at distinct communication rounds, aiming to minimize the overall communication burden. Finally, for users with data sets of similar distributions, the model updates may be statistically correlated, indicating the possibility to exploit this correlation to further reduce the communication load via, e.g., distributed source coding techniques.

Furthermore, the fact that model updates are often compressed in a manner that induces distortion motivates the analysis of lossy compression as a privacy-preserving mechanism. In particular, differential privacy amplification methods rely on adding artificial distortion to the model updates, where the motivation is to obscure the updates such that each data sample used in their training cannot be recovered. Since quantization and lossy compression already induce some level of distortion, one can reduce the level of artificial noise needed to guarantee differential privacy. Furthermore, the fact that quantization discretizes the model updates facilitates applying encryption methods, which often assume integer quantities.

### *Uplink transmission*

The main challenge in uplink transmission stems from the fact that communication is often carried out over shared noisy channels that are limited in their spectral and temporal resources. As discussed in the "Learning-Aware Resource Allocation" section, the delay induced by communication and its harmful effect on the convergence time and the accuracy of a model learned in a federated manner can be significantly reduced by dividing the channel resources in a learning-aware fashion, for instance, by boosting interference-free orthogonal communications while prioritizing users that are expected to have a greater contribution to the learning process and even by promoting full reuse via over-the-air computation.

Nonetheless, there is still much to be explored in the study of uplink transmission schemes for federated learning. In particular, while user selection methods give all devices a chance to participate, the prioritization among the users should account for a multitude of considerations that are not accounted for to date. For instance, the fact that edge devices are comprised of a broad range of different technologies indicates that some devices are more likely to transmit at increased delays. Statistical heterogeneity also plays an important role, particularly when one has prior knowledge of the distribution of each user, as in clustered federated learning [49]. Here, one should consider whether it is preferable to assign resources and select users in a diverse or homogeneous

## Byzantine–Robust Federated Learning

Consider a federated learning system with $N$ users possessing data sets of identical sizes; i.e., $|\mathcal{D}^i| \equiv n$. Let $\alpha$ denote the fraction of the $N$ unreliable users, while the remaining $1-\alpha$ fraction is normal, and denote the model updates by $\boldsymbol{g}_t^i = [g_{t,1}^i, \ldots g_{t,d}^i] \triangleq \boldsymbol{\theta}_t^i - \boldsymbol{\theta}_{t-E}^i$. In each communication round, the model updates are sent to the server for aggregation, where the Byzantine users can send arbitrary messages; in particular, they may have complete knowledge of the system and learning algorithms and can collude with one another. Assume an orthogonal transmission over a noiseless channel, and let $\boldsymbol{y}_t$ be the channel output that contains the received model updates; i.e., $\boldsymbol{y}_t = [\boldsymbol{g}_t^1, \ldots, \boldsymbol{g}_t^N]^T$.

Two combining strategies proposed in [57] to address malicious users are the coordinate-wise median and coordinate-wise trimmed mean. The coordinate-wise median method aggregates the local updates using 1D median

$$\boldsymbol{g}_t(\boldsymbol{\theta}) = \text{med}\{\boldsymbol{y}_t\}, \tag{S9}$$

where $\boldsymbol{g}_t$ is a vector, with its $k$ coordinate being $g_{t,k} = \text{med}\{g_{t,k}^i : i \in \mathcal{N}\}$ for each $k \in \{1 \ldots d\}$. The coordinate-wise $\beta$-trimmed mean method removes the largest and smallest $\beta$ fraction of the $k$th coordinate of the model update vector; i.e.,

$$g_{t,k} = \text{trmean}_\beta\{\boldsymbol{y}_{t,k}\} = \frac{1}{(1-2\beta)N} \sum_{g \in U_k} g, \tag{S10}$$

where $U_k$ is a subset of $\{g_{t,k}^1, \ldots g_{t,k}^N\}$ obtained by removing the largest and smallest $\beta$ fraction of its elements. Typically, $\beta$ is set to be no smaller than the expected portion of unreliable users, which is denoted by $\alpha$. The two strategies are designed to reduce the effects of possible outliers before the global aggregation. Both aggregation strategies (S9) and (S10) have provable guarantees on the convergence rates.

The authors of [57] characterized the accuracy of such Byzantine–robust aggregation, focusing on setups where each user sends the loss gradients, i.e., $E = 1$, and the loss surface is strongly convex and smooth and with bounded skewness. It was shown in [57] that in such cases, the expected gap between the achieved loss function using the coordinate-wise median method and the minimum loss is an increasing function of $\alpha$ and a decreasing function of $n$ (specifically, with rate $1/n$). For strongly convex, smooth objectives and subexponential gradients, and by setting $\beta \geq \alpha$, the expected gap between the achieved loss function using the coordinate-wise $\beta$-trimmed mean method and the minimum loss is an increasing function of $\alpha$ and a decreasing function of $n$ (specifically, with rate $1/\sqrt{n}$). Although the median-based algorithm has a poorer convergence rate with respect to the parameter $n$, it requires milder tail/moments assumptions and does not need the knowledge of the maximal number of expected unreliable users, encapsulated in the portion $\alpha$. The performance of the two methods under different values of $\alpha$ and loss functions is demonstrated in Figure S4.

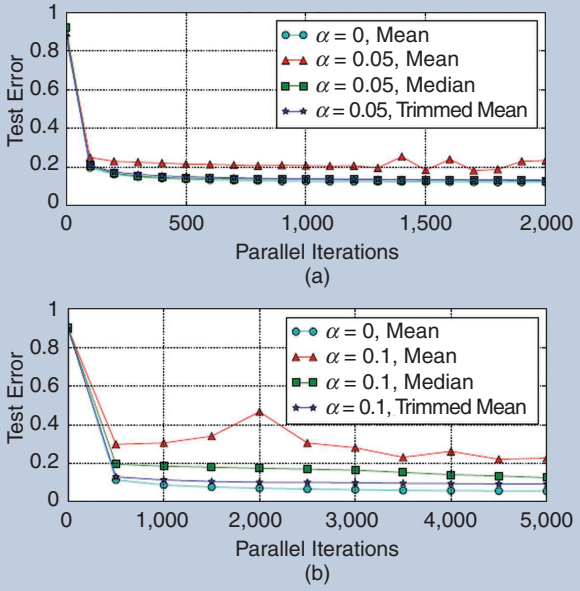



**FIGURE S4.** A numerical study of the coordinate-wise median and trimmed-mean methods, adapted from [57] with the authors' permission. The considered scenario is handwritten digit classification based on the Modified National Institute of Standards and Technology data set distributed in an independent identically distributed fashion between $N = 40$ users in the logistic regression model and $N = 10$ in the convolutional neural network model. The methods are compared to vanilla distributed gradient descent (aggregating the gradients by taking the mean), where the trimmed mean uses $\beta = \alpha$. (a) Logistic regression. (b) A convolutional neural network.

fashion. Finally, when privacy-enhancing mechanisms, such as those based on artificial noise, are used, one can design user selection schemes and account for their presence by, e.g., prioritizing users for which a lesser amount of artificial noise is needed to meet a given privacy constraint.

Furthermore, resource allocation and user selection can contribute to nonorthogonal transmissions. Here, it is possible to rely on dynamic spectrum access schemes, which are commonly studied for wireless sensor networks. Developing dynamic spectrum access strategies for federated learning tasks is subject to challenges that are not encountered in traditional domains. For instance, intriguing research questions are 1) how to develop dynamic spectrum access that integrates over-the-air and orthogonal transmissions when coherent over-the-air transmission over the entire network is not efficient and 2) how to develop spatiotemporal dynamic spectrum access strategies when users experience deep fading in time and frequency. For example, a promising direction is to model the fading channel as a known or unknown Markovian process, for which theoretical performance measures for channel allocation strategies can be developed rigorously, as in, e.g., [59].

Finally, we note that the vast majority of the existing federated learning algorithms focus on communications carried out over networks obeying a star topology, in which all users communicate directly with the server. Nevertheless, common communication schemes in wireless networks allow multihop communications between edge devices via wireless access points, relays, or even using the emerging technology of reconfigurable intelligent surfaces. In such cases, different subsets of the users may share distinct channels, while communications on both the uplink and the downlink are carried out over a multihop route. Learning in a federated manner over hierarchical communication networks requires dedicated treatment regarding the division of channel resources, the presence of intermediate aggregations, and the possibilities of model caching along the routes as well as carrying out over-the-air federated learning in a hierarchical manner.

### Global combining

*Global combining* refers to how the channel output observed by the server orchestrating the federated learning procedure is translated into an updated global model. Applying combining mechanisms other than the conventional FedAvg rule (6) enables us to compensate for distortions induced in the encoding and transmission of the model updates, mitigate the harmful effect of unreliable and malicious users, and facilitate accurate inference in the presence of heterogeneous data division.

The fact that most nonsecure combining mechanisms for tackling heterogeneity rely on weighted averaging motivates exploring nonlinear combinations. Such an analysis can be combined with Byzantine–robust aggregation, which inherently utilizes nonlinear mappings to omit outliers, exploring the joint functionality in training a model capable of accurately inferring and using each of the individual distributions employed for its training, possibly yielding personalized models [52], while being tolerant to the presence of malicious users.

Furthermore, the heterogeneous nature of the data and the users in federated learning systems implies that mechanisms for truncating the contribution of unreliable users must account for such properties. One approach to do so is to divide the users into clusters based on their distribution and implement Byzantine–robust aggregation separately over each of these homogeneous clusters, as proposed in [51]. The fact that such a strategy already trains multiple models for each cluster motivates its combination with aggregation mechanisms that combine the inference rules during testing.

### Additional directions

Our description so far follows the division of the global aggregation stage in the federated learning flow into the aforementioned encoding, transmission, and combining stages. This structured framework enables identifying the relationships between federated learning schemes derived for different purposes, and it facilitates the derivation of dedicated future methods. Nonetheless, one is not limited to design mechanisms within this framework, and it is natural to also consider the joint optimization of multiple steps. Furthermore, while most of the key challenges of federated learning that are natural to address using signal processing techniques lie in the global aggregation stage, the federated learning flow is also comprised of the model distribution and local training steps. It is thus of interest to extend the scope beyond the aggregation of the model updates into a global model.

One such research direction involves the joint optimization of the multiple stages of which global aggregation is comprised. For instance, jointly treating the local encoding and uplink transmission stages can be studied as a distributed joint-source-channel coding setup, possibly minimizing the delay in conveying the overall model updates. In addition, it is likely that one can further benefit in terms of convergence speed by designing such direct mapping from model updates into the channel input of the complete set of users in a joint manner with the combining mechanism at the server side.

Additional relevant research directions that go beyond the division considered in this article focus on reducing the communication overload and boosting privacy in the presence of external adversaries during downlink transmission. Another possible direction is to improve the convergence time in the local training stage, without modifying the optimization algorithm by properly selecting the data. Here, one can consider choosing a subset of the data used for training via active learning techniques, which dates back to Chernoff's framework of optimal experimental design [50], to reduce the training time and thus the overall convergence delay. Therefore, an intriguing question is how to select the useful data among multiple users in the heterogeneous federated learning setting. This will allow significant bandwidth and energy savings by prioritizing informative data sources. Another research direction is to combine federated learning with deep network designs that support compact architectures that are suitable for the limited computational and data resources of edge devices. One such candidate is the framework of model-based deep learning, which designs

interpretable, compact networks based on domain knowledge [60], [61]. The combination of model-based deep learning designs with federated learning treats the paradigm of deep learning on the edge in terms of both architecture and training in a manner that is oriented toward the capabilities and limitations of edge devices.

Finally, while federated learning deals with the training of a machine learning model, its distributed nature also motivates the study of how to use this learned model during inference. For instance, federated learning carried out in a clustered fashion [51], [55] results in different users having access to various models as well as a centralized server that has access to all these representations. In this case, one can consider scenarios in which multiple edge users collaborate during inference as a form of the wisdom of the crowd, as proposed in [53], or alternatively, combining server-based and localized inference to enable personalized decision making.

## Acknowledgment

This research was partially supported by the Israel Science Foundation (grant 2640/20).

## Authors

***Tomer Gafni*** (gafnito@post.bgu.ac.il) received his M.Sc. degree in electrical and computer engineering from Ben Gurion University of the Negev in 2020. He is currently working toward his Ph.D. degree at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 8410501, Israel. His research interests include sequential learning, decision theory, and statistical inference and learning, with applications in large-scale systems and wireless networks.

***Nir Shlezinger*** (nirshlezinger1@gmail.com) received his Ph.D. degree in electrical and computer engineering from Ben-Gurion University of the Negev in 2017. He is currently an assistant professor at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 8410501, Israel. From 2017 to 2019, he was a postdoctoral researcher at Technion, Israel Institute of Technology, and from 2019 to 2020, he was a postdoctoral researcher in the Signal Acquisition Modeling, Processing, and Learning Lab, Weizmann Institute of Science. His research interests include the intersection of communications, signal processing, information theory, and machine learning.

***Kobi Cohen*** (kobi.cohen10@gmail.com) received his Ph.D. degree in electrical engineering from Bar-Ilan University, Ramat Gan, Israel, in 2013. He is a senior lecturer (tenured assistant professor) at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 8410501, Israel. His research interests include decision theory, stochastic optimization, and statistical inference and learning, with applications and analysis in communication networks and cyber- and large-scale systems. He received various awards, including the Best Paper Award at the 2015 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks; 2011 Feder Family Award (second prize), Advanced Communication Center, Tel Aviv University; and President Fellowship from 2008 to 2012 and top honor list prizes from Bar-Ilan University in 2006, 2010, and 2011, respectively. He is a Senior Member of IEEE.

***Yonina C. Eldar*** (yonina.eldar@weizmann.ac.il) received her Ph.D. degree in electrical engineering and computer science in 2002 from the Massachusetts Institute of Technology, Cambridge. She is a professor in the Department of Math and Computer Science, Weizmann Institute of Science, Rehovot, 7610001, Israel, where she heads the Center for Biomedical Engineering and Signal Processing. She is also a visiting professor at the Massachusetts Institute of Technology and the Broad Institute as well as an adjunct professor at Duke University. She is a member of the Israel Academy of Sciences and Humanities and a European Association for Signal Processing fellow. She has received awards including the IEEE Signal Processing Society Technical Achievement Award, IEEE Radar Systems Panel/IEEE Aerospace and Electronic Systems Society Fred Nathanson Memorial Radar Award, and IEEE Kiyo Tomiyasu Award. Her research interests are in the broad areas of statistical signal processing, sampling theory and compressed sensing, learning and optimization methods, and their applications to biology and optics. She is a Fellow of IEEE.

***H. Vincent Poor*** (poor@princeton.edu) received his Ph.D. degree in electrical engineering and computer science from Princeton University in 1977. He is currently the Michael Henry Strater University Professor at Princeton University, Princeton, New Jersey, 08544, USA. He is a member of the U.S. National Academy of Engineering and U.S. National Academy of Sciences and a foreign member of the Chinese Academy of Sciences and the Royal Society. He received the Technical Achievement and Society Awards from the IEEE Signal Processing Society in 2007 and 2011, respectively, and the IEEE Alexander Graham Bell Medal in 2017, and the IEEE Alexander Graham Bell Medal in 2017. His research interests include information theory, machine learning, and network science and their applications in wireless networks, energy systems, and related fields.

## References

[1] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019, doi: 10.1109/JPROC.2019.2921977.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statistics*, 2017, pp. 1273–1282.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles et al., "Advances and open problems in federated learning," 2019, arXiv:1912.04977.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020, doi: 10.1109/MSP.2020.2975749.

[5] L. Li, Y. Fan, M. Tse, and K. Y. Lin, "A review of applications in federated learning," *Comput. Ind. Eng.*, vol. 149, p. 106854, 2020, doi: 10.1016/j.cie.2020.106854.

[6] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learning Representations*, 2018.

[7] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 1709–1720.

[8] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 500–514, Dec. 23, 2020, doi: 10.1109/TSP.2020.3046971.

[9] S. Zheng, C. Shen, and X. Chen, "Design and analysis of uplink and downlink communications for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2150–2167, 2021, doi: 10.1109/JSAC.2020.3041388.

[10] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, 2019, doi: 10.1109/TWC.2019.2946245.

[11] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, 2020, doi: 10.1109/TWC.2020.2974748.

[12] T. Sery and K. Cohen, "On analog gradient descent learning over multiple access fading channels," *IEEE Trans. Signal Process*, vol. 68, pp. 2897–2911, Apr. 22, 2020, doi: 10.1109/TSP.2020.2989580.

[13] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proc. Nat. Academy Sci.s U.S.A*, vol. 118, no. 17, 2021, doi: 10.1073/pnas.2024789118.

[14] C. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, 2021, doi: 10.1109/TNET.2020.3035770.

[15] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FEDAVG on non-IID data," in *Proc. Int. Conf. Learning Representations*, 2020.

[16] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inform. Process. Syst.*, 2019.

[17] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 80–94, 2004, doi: 10.1109/MSP.2004.1328091.

[18] S.-H. Tsai and H. V. Poor, "Power allocation for artificial-noise secure MIMO precoding systems," *IEEE Trans. Signal Process.*, vol. 62, no. 13, pp. 3479–3493, 2014, doi: 10.1109/TSP.2014.2329273.

[19] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4452–4463.

[20] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, arXiv:1610.05492.

[21] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. 35th Int. Conf. Machine Learning*, 2018, pp. 560–569.

[22] R. M. Gray and T. G. Stockham, "Dithered quantizers," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 805–812, 1993, doi: 10.1109/18.256489.

[23] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge, MA, USA: Cambridge Univ. Press, 2012.

[24] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, Mar. 2020, doi: 10.1109/TSP.2020.2981904.

[25] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP J. Inform. Security*, vol. 2007, no. 1, pp. 1–10, 2007, doi: 10.1155/2007/13801.

[26] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found.s Trends Theoretical Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014, doi: 10.1561/0400000042.

[27] Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, 2017, doi: 10.1109/TIFS.2017.2787987.

[28] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1175–1191.

[29] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020, doi: 10.1109/TIFS.2020.2988575.

[30] D. Liu and O. Simeone, "Privacy for free: Wireless federated learning via uncoded transmission with adaptive power control," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 170–185, 2020, doi: 10.1109/JSAC.2020.3036948.

[31] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security*, 2019, doi: 10.1145/3338501.3357370.

[32] F. Meshkati, H. V. Poor, and S. C. Schwartz, "Energy-efficient resource allocation in wireless networks," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 58–68, 2007, doi: 10.1109/MSP.2007.361602.

[33] W. Liu, X. Zang, Y. Li, and B. Vucetic, "Over-the-air computation systems: Optimization, analysis and scaling laws," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5488–5502, 2020, doi: 10.1109/TWC.2020.2993703.

[34] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019, arXiv:1902.01046.

[35] 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 Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, 2021, doi: 10.1109/TWC.2020.3024629.

[36] R. Balakrishnan, M. Akdeniz, S. Dhakal, A. Anand, A. Zeira, and N. Himayat, "Resource management and model personalization for federated learning over wireless edge networks," *J. Sensor Actuator Netw.*, vol. 10, no. 1, p. 17, 2021, doi: 10.3390/jsan10010017.

[37] W. Xia, T. Q. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, 2020, doi: 10.1109/TWC.2020.3008091.

[38] Y. Song, H. Chang, Z. Zhou, S. Jere, and L. Liu, "Federated dynamic spectrum access," 2021, arXiv:2106.14976.

[39] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3498–3516, 2007, doi: 10.1109/TIT.2007.904785.

[40] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Comput.*, vol. 8, no. 3, pp. 643–674, 1996, doi: 10.1162/neco.1996.8.3.643.

[41] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3863–3877, 2013, doi: 10.1109/TCOMM.2013.072913.120815.

[42] O. Abari, H. Rahul, D. Katabi, and M. Pant, "Airshare: Distributed coherent transmission made seamless," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 1742–1750, doi: 10.1109/INFOCOM.2015.7218555.

[43] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "Over-the-air federated learning from heterogeneous data," *IEEE Trans. Signal Process.*, vol. 69, pp. 3796–3811, 2021, doi: 10.1109/TSP.2021.3090323.

[44] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, 2020, doi: 10.1109/TWC.2019.2961673.

[45] M. M. Amiri, T. M. Duman, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Blind federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5129–5143, 2021, doi: 10.1109/TWC.2021.3065920.

[46] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. 36th Int. Conf. Machine Learning (ICML 2019)*, 2019, pp. 8114–8124.

[47] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Multiple source adaptation and the Rényi divergence," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 367–374.

[48] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th {USENIX} Security Symp. ({USENIX} Security 20)*, 2020, pp. 1605–1622.

[49] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 3710–3722, 2020.

[50] H. Chernoff, "Sequential design of experiments," *Ann. Math. Statist.*, vol. 30, no. 3, pp. 755–770, 1959, doi: 10.1214/aoms/1177706205.

[51] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," 2019, arXiv:1906.06629.

[52] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, arXiv:2002.10619.

[53] N. Shlezinger, E. Farhan, H. Morgenstern, and Y. C. Eldar, "Collaborative inference via ensembles on the edge," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 2021, pp. 8478-8482, doi: 10.1109/ICASSP39728.2021.9414740.

[54] S. Lee, C. Park, S.-N. Hong, Y. C. Eldar, and N. Lee, "Bayesian federated learning over wireless networks," 2020, arXiv:2012.15486.

[55] N. Shlezinger, S. Rini, and Y. C. Eldar, "The communication-aware clustered federated learning problem," in *Proc. IEEE Int. Symp. Inform. Theory*, 2020, pp. 2610–2615, doi: 10.1109/ISIT44484.2020.9174245.

[56] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Measurement Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017, doi: 10.1145/3154503.

[57] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Machine Learning*, 2018, pp. 5650–5659.

[58] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant SGD," 2018, arXiv:1802.10116.

[59] T. Gafni and K. Cohen, "Distributed learning over Markovian fading channels for stable spectrum access," 2021, arXiv:2101.11292.

[60] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," 2020, arXiv:2012.08405.

[61] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, 2021, doi: 10.1109/MSP.2020.3016905.