Federated Learning in Mobile Edge Networks: A Comprehensive Survey

Preprint · September 2019

CITATIONS
0

READS
462

8 authors, including:

Cong Luong Nguyen
Hanoi Junior High School Vietnam
30 PUBLICATIONS **364** CITATIONS
SEE PROFILE

Hoang Dinh Thai
Nanyang Technological University
78 PUBLICATIONS **2,652** CITATIONS
SEE PROFILE

Yutao Jiao
Nanyang Technological University
16 PUBLICATIONS **134** CITATIONS
SEE PROFILE

Ying-Chang Liang
University of Electronic Science and Technology of China
546 PUBLICATIONS **20,476** CITATIONS
SEE PROFILE

Federated Learning in Mobile Edge Networks: A Comprehensive Survey

Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, *Fellow, IEEE*, Qiang Yang, *Fellow, IEEE*, Dusit Niyato, *Fellow, IEEE*, and Chunyan Miao

***Abstract*—In recent years, mobile devices are equipped with increasingly advanced sensing and computing capabilities. Coupled with advancements in Deep Learning (DL), this opens up countless possibilites for meaningful applications, e.g., for medical purposes and in vehicular networks. Traditional cloud-based Machine Learning (ML) approaches require the data to be centralized in a cloud server or data center. However, this results in critical issues related to unacceptable latency and communication inefficiency. To this end, Mobile Edge Computing (MEC) has been proposed to bring intelligence closer to the edge, where data is produced. However, conventional enabling technologies for ML at mobile edge networks still require personal data to be shared with external parties, e.g., edge servers. Recently, in light of increasingly stringent data privacy legislations and growing privacy concerns, the concept of Federated Learning (FL) has been introduced. In FL, end devices use their local data to train an ML model required by the server. The end devices then send the model updates rather than raw data to the server for aggregation. FL can serve as an enabling technology in mobile edge networks since it enables the collaborative training of an ML model and also enables DL for mobile edge network optimization. However, in a large-scale and complex mobile edge network, heterogeneous devices with varying constraints are involved. This raises challenges of communication costs, resource allocation, and privacy and security in the implementation of FL at scale. In this survey, we begin with an introduction to the background and fundamentals of FL. Then, we highlight the aforementioned challenges of FL implementation and review existing solutions. Furthermore, we present the applications of FL for mobile edge network optimization. Finally, we discuss the important challenges, open issues and future research directions in FL.**

***Index Terms*—Federated Learning, mobile edge networks, resource allocation, communication cost, data privacy, data security**

I. Introduction

Currently, there are nearly 7 billion connected Internet of Things (IoT) devices [1] and 3 billion smartphones around the world. These devices are equipped with increasingly advanced sensors, computing, and communication capabilities. As such, they can potentially be deployed for various crowdsensing tasks, e.g., for medical purposes [2] and air quality monitoring [3]. Coupled with the rise of Deep Learning (DL) [4], the wealth of data collected by end devices opens up countless possibilities for meaningful research and applications.

In the traditional cloud-centric approach, data collected by mobile devices is uploaded and processed centrally in a cloud-based server or data center. In particular, data collected by IoT devices and smartphones such as measurements [5], photos [6], videos [7], and location information [8] are aggregated at the data center [9]. Thereafter, the data is used to provide insights or produce effective inference models. However, this approach is no longer sustainable for the following reasons. Firstly, data owners are increasingly privacy sensitive. Following privacy concerns among consumers in the age of big data, policy makers have responded with the implementation of data privacy legislations such as the European Commission's General Data Protection Regulation (GDPR) [10] and Consumer Privacy Bill of Rights in the US [11]. In particular, the consent (GDPR Article 6) and data minimalization principle (GDPR Aricle 5) limits data collection and storage only to what is consumer-consented and absolutely necessary for processing. Secondly, a cloud-centric approach involves long propagation delays and incurs unacceptable latency [12] for applications in which real-time decisions have to be made, e.g., in self-driving car systems [13]. Thirdly, the transfer of data to the cloud for processing burdens the backbone networks especially in tasks involving unstructured data, e.g., in video analytics [14]. This is exacerbated by the fact that cloud-centric training is relatively reliant on wireless communications [15]. As a result, this can potentially impede the development of new technologies.

With data sources mainly located outside the cloud today [16], Mobile Edge Computing (MEC) has naturally been proposed as a solution in which the computing and storage capabilities [12] of end devices and edge servers are leveraged on to bring model training closer to where data is produced [17]. As defined in [15], an end-edge-cloud computing network comprises: (i) end devices, (ii) edge nodes, and (iii) cloud server. For model training in conventional MEC approaches, a collaborative paradigm has been proposed in which training data are first sent to the edge servers for model training up to lower level DNN layers, before computation intensive tasks are offloaded to the cloud [18], [19] (Fig. 1). However, this arrangement incurs significant communication costs and is unsuitable especially for applications that require persistent training [15]. In addition, computation offloading and data

W. Y. B. Lim is with Alibaba-NTU Joint Research Institute, Nanyang Technological University, Singapore. Email: limw0201@e.ntu.edu.sg.

N. C. Luong, Y. Jiao, D. Niyato and C. Miao are with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mails: clnguyen@ntu.edu.sg, yjiao001@e.ntu.edu.sg, dniyato@ntu.edu.sg, ascymiao@ntu.edu.sg.

D. T. Hoang is with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. E-mail: hoang.dinh@uts.edu.au.

Y.-C. Liang is with the Center for Intelligent Networking and Communications (CINC), University of Electronic Science and Technology of China (UESTC), Chengdu 611731, China. E-mail: liangyc@ieee.org.

Q. Yang is with Hong Kong University of Science and Technology, Hong Kong, China. Email: qyang@cse.ust.hk.

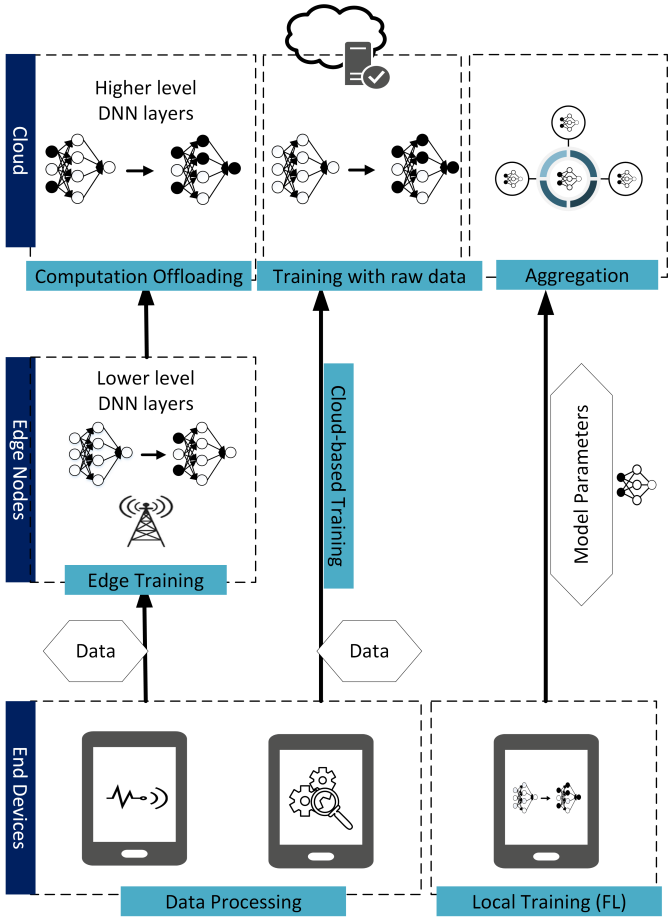

Fig. 1: Edge AI approach brings AI processing closer to where data is produced. In particular, FL allows training on devices where the data is produced.

processing at edge servers still involve the transmission of potentially sensitive personal data. This can discourage privacy-sensitive consumers from taking part in model training, or even violate increasingly stringent privacy laws [10]. Although various privacy preservation methods, e.g., differential privacy (DP) [20], have been proposed, a number of users are still not willing to expose their private data for fear that their data may be inspected by external servers. In the long run, this discourages the development of technologies as well as new applications.

To guarantee that training data remains on personal devices and to facilitate collaborative machine learning of complex models among distributed devices, a decentralized ML approach called Federated Learning (FL) is introduced in [21]. In FL, mobile devices use their local data to cooperatively train an ML model required by an FL server. They then send the model updates, i.e., the model's weights, to the FL server for aggregation. The steps are repeated in multiple rounds until a desirable accuracy is achieved. This implies that FL can be an enabling technology for ML model training at mobile edge networks. As compared to conventional cloud-centric ML model training approaches, the implementation of FL for model training at mobile edge networks features the following advantages.

- *Highly efficient use of network bandwidth:* Less information is required to be transmitted to the cloud. For example, instead of sending the raw data over for processing, participating devices only send the updated model parameters for aggregation. As a result, this significantly reduces costs of data communication and relieves the burden on backbone networks.
- *Privacy:* Following the above point, the raw data of users need not be sent to the cloud. This guarantees user privacy. In fact, with guaranteed privacy, more users will be willing to take part in collaborative model training and so, better inference models are built.
- *Low latency:* With FL, ML models can be consistently trained and updated. Meanwhile, in the MEC paradigm, real-time decisions, e.g., event detection [22], can be made locally at the edge nodes or end devices. Therefore, the latency is much lower than that when decisions are made in the cloud before transmitting them to the end devices. This is vital for time critical applications such as self-driving car systems in which the slightest delays can potentially be life threatening [13].

Given the aforementioned advantages, FL has seen recent successes in several applications. For example, the Federated Averaging algorithm (*FedAvg*) proposed in [23] has been applied to Google's Gboard [24] to improve next-word prediction models. In addition, several studies have also explored the use of FL in a number of scenarios in which data is sensitive in nature, e.g., to develop predictive models for diagnosis in health AI [25] and to foster collaboration across multiple Government agencies [26].

In addition, besides being an enabling technology for ML model training *at* mobile edge networks, FL is also an enabling technology *for* mobile edge network optimization. Given the computation and storage constraints of increasingly complex mobile edge networks, conventional network optimization approaches that are built on static models fare relatively poorly in modelling dynamic networks [15]. As such, a data-driven Deep Learning (DL) based approach [27] for optimizing resource allocation is increasingly popular. For example, DL can be used for representation learning of network conditions [28] whereas Deep Reinforcement Learning (DRL) can optimize decision making through interactions with the dynamic environment [29]. However, the aforementioned approaches require user data as an input and these data may be sensitive or inaccessible in nature due to regulatory constraints. As such, in this survey, we also consider FL's potential to serve as an enabling technology for optimizing mobile edge networks, e.g., in cell association [30], computation offloading [31], and vehicular networks [32].

However, there are several challenges to implementing FL at scale. Firstly, due to the high dimensonality of model updates and limited communication bandwith of participating mobile devices, communication costs remain an issue. Secondly, in a large and complex mobile edge network, the heterogeneity of participating devices in terms of data quality, computation power, and willingness to participate have to be well managed from the resource allocation perspective. Thirdly, recent research works have clearly shown that a malicious

participant may exist in FL and can infer the information of other participants from shared parameters. As such, privacy and security issues in FL need to be considered.

Although there are surveys on MEC and FL, the existing studies usually treat the two topics separately. For existing surveys on FL, the authors in [33] place more emphasis on discussing the architecture and categorization of different FL settings to be used for the varying distributions of training data. The authors in [34] highlight the applications of FL in wireless communications but do not discuss the issues pertaining to FL implementation. In addition, the focus of [34] is on cellular network architecture rather than mobile edge networks. In contrast, the authors in [35] provide a brief tutorial on FL and the challenges related to its implementation, but do not consider the issue of resource allocation in FL, or the potential applications of FL for mobile edge network optimization. On the other hand, for surveys in MEC that focus on implementing ML model training at edge networks, a macroscopic approach is usually adopted in which FL is briefly mentioned as one of the enabling technologies in the MEC paradigm, but without detailed elaboration with regards to its implementation or the related challenges. In particular, the authors in [14], [36], and [37] study the architectures and process of training and inference at edge networks without considering the challenges to FL implementation. In addition, surveys studying the implementation of DL for mobile edge network optimization mostly do not focus on FL as a potential solution to preserve data privacy. For example, the authors in [19], [38], [39] and [40] discuss strategies for optimizing computation offloading for mobile edge networks, but do not consider the use of privacy preserving federated approaches in their studies. Similarly, [29] considers the use of DRL in communications and networking but do not include federated DRL approaches.

In summary, most existing surveys on FL do not consider the applications of FL in the context of mobile edge networks, whereas existing surveys on MEC do not consider the challenges to FL implementation, or the potential of FL approaches in mobile edge network optimization. This motivates us to have a comprehensive survey that covers: (i) a tutorial on FL implementation (ii) unique features of FL and the resulting implementation challenges and (iii) FL as an enabling technology for mobile edge network optimization. For the reader's convenience, we classify the related studies to be discussed in this survey in Fig. 2. The classification is based on (i) FL at mobile edge network, i.e., studies that focus on the challenges of implementing collaborative training of ML models on end devices and (ii) FL for mobile edge network, i.e., studies that explore the use of FL for mobile edge network optimization. We also present a list of common abbreviations for reference in Table I.

The rest of this paper is organized as follows. Section II introduces the background and fundamentals of FL. Section III reviews solutions provided to reduce communication costs. Section IV discusses resource allocation approaches in FL. Section V discusses privacy and security issues. Section VI discusses applications of FL for mobile edge network optimization. Section VII discusses the challenges, open issues and future research directions in FL. Section VIII concludes the paper.

TABLE I: List of common abbreviations.

Abbreviation	Description
BAA	Broadband Analog Aggregation
CNN	Convolutional Neural Network
CV	Computer Vision
DDQN	Double Deep Q-Network
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential Privacy
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
FedAvg	Federated Averaging
FL	Federated Learning
GAN	Generative Adversarial Network
IID	Independent and Identically Distributed
IoT	Internet of Things
IoV	Internet of Vehicles
LSTM	Long Short Term Memory
MEC	Mobile Edge Computing
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
OFDMA	Orthogonal Frequency-division Multiple Access
QoE	Quality of Experience
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMPC	Secure Multiparty Computation
SNR	Signal-to-noise ratio
SVM	Support Vector Machine
TFF	TensorFlow Federated
UE	User Equipment
URLLC	Ultra reliable low latency communication

II. Background and Fundamentals of Federated Learning

Artificial Intelligence (AI) has become an essential part of our lives today, following the recent successes and progression of DL in several domains, e.g., Computer Vision (CV) [41] and Natural Language Processing (NLP) [42]. In traditional training of Deep Neural Networks (DNNs), a cloud based approach is adopted whereby data is centralized and model training occurs in powerful cloud servers. However, given the ubiquity of mobile devices that are equipped with increasingly advanced sensing and computing capabilities, the trend of migrating intelligence from the cloud to the edge, i.e., in the MEC paradigm, has naturally arisen. In addition, amid growing privacy concerns, the concept of FL has been proposed.

FL involves the collaborative training of DNN models on end devices. There are, in general, two steps in the FL training process namely (i) local model training and (ii) global aggregation of updated parameters. In this section, we first provide a brief introduction to DNN model training, which generalizes local model training in FL. Note that while FL can be applied to the training of ML models in general, we focus specifically on DNN model training in this section for two reasons. Firstly, the implementation of FL at mobile edge networks can naturally leverage on the increasing computing power and wealth of data collected by distributed end devices, both of which are driving forces contributing to the rise of DL

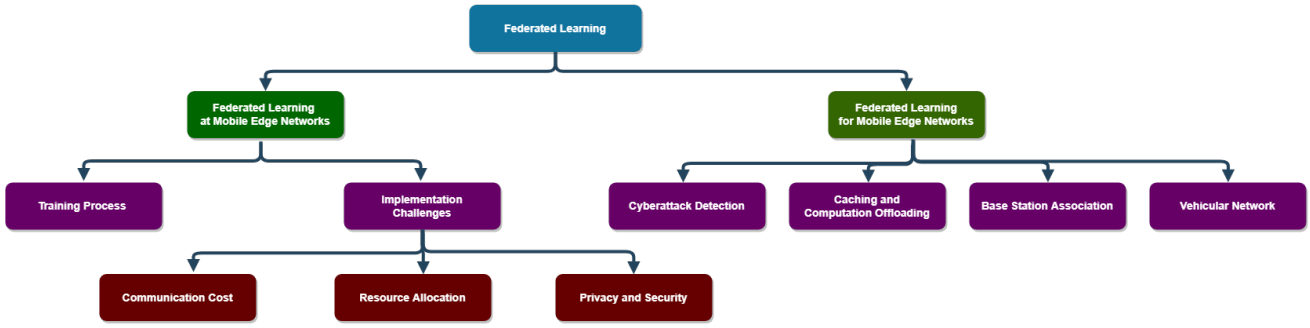

Fig. 2: Classification of related studies to be discussed in this survey.

[43]. Secondly, a majority of the papers that we review focus on the federated training of DNN models. As such, a brief introduction to general DNN model training will be useful for subsequent sections. Thereafter, we proceed to provide a tutorial of the FL training process that incorporates both global aggregation and local training. In addition, we also highlight the statistical challenges of FL model training and present the protocols and open-source frameworks of FL.

A. Deep Learning

Conventional ML algorithms rely on *hand-engineered* feature extractors to process raw data [44]. As such, domain expertise is often a prerequisite for building an effective ML model. In addition, feature selection has to be customized and reinitiated for each new problem. On the other hand, DNNs are representation learning based, i.e., DNNs can automatically discover and learn these features from raw data [4] and thus often outperform conventional ML algorithms especially when there is an abundance of data.

DL lies within the domain of the brain-inspired computing paradigm, of which the neural network is an important part of [45]. In general, a neural network design emulates that of a neuron [46]. It comprises three layers: (i) input layer, (ii) hidden layer, and (iii) output layer. In a feedforward neural network, a weighted and bias-corrected input value is passed through a non-linear activation function to derive an output [47] (Fig. 3). Some activation functions include the ReLu and softmax functions [42]. A typical DNN comprises multiple hidden layers that map an input to an output. For example, the goal of a DNN trained for image classification [48] is to produce a vector of scores as the output, in which the positional index of the highest score corresponds to the class to which the input image is classified to belong. As such, the objective of training a DNN is to optimize the weights of the network such that the loss function, i.e., difference between the ground truth and model output, is minimized.

Before training, the dataset is first split into the training and inference dataset. Then, the training dataset is used as input data for the optimization of weights in the DNN. The weights are calibrated through stochastic gradient descent (SGD), in which the weights are updated by the product of (i) the learning rate lr, i.e., how fast the weights are updated per iteration, and (ii) partial derivative of the loss function L with respect to the weight w. The SGD formula is as follows:

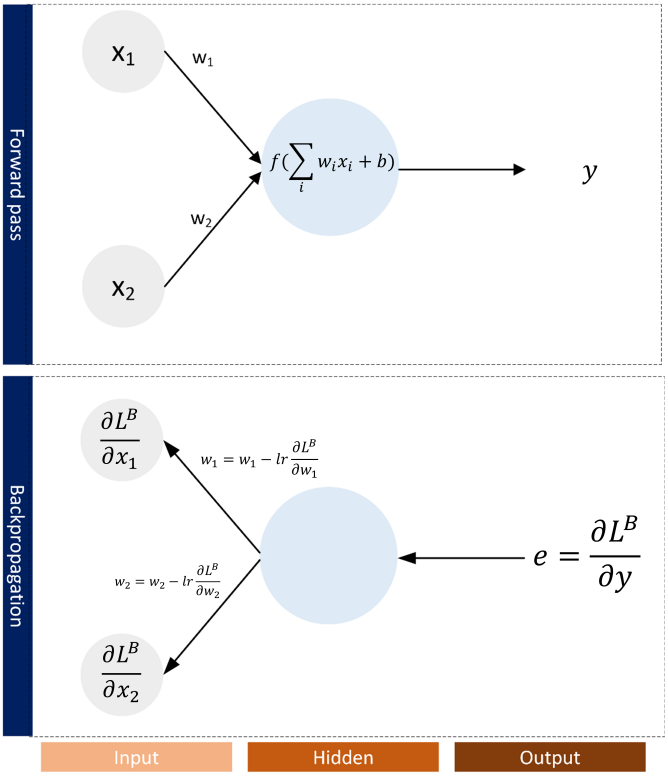

Fig. 3: In forward pass, an output is derived from the weights and inputs. In backpropagation, the input gradient e is used to calibrate the weights of the DNN model.

$$W = W - lr\frac{\partial L}{\partial W} \tag{1}$$

$$\frac{\partial L}{\partial W} \approx \frac{1}{m}\sum_{i \epsilon B}\frac{\partial l^{(i)}}{\partial W} \tag{2}$$

Note that the SGD formula presented in (1) is that of a mini-batch GD. In particular, equation (2) is derived as the average gradient matrix over the gradient matrices of B batches, in which each batch is a random subset consisting of m training samples. This is preferred over the full batch GD, i.e., where the entirety of the training set is included in computing the partial derivative, since the full batch GD can lead to slow training and batch memorization [49]. The gradient matrices are derived through backpropagation from the input gradient

e (Fig. 3) [46].

The training iterations are then repeated over many epochs, i.e., full passes over the training set, for loss minimalization. A well-trained DNN generalizes well, i.e., achieve high *inference* accuracy when applied to data that it has not seen before, e.g., the test set. There are other alternatives to supervised learning, e.g., semi-supervised learning [50], unsupervised learning [51] and reinforcement learning [52], as well as several DNN architectures tailored to suit varying functions, e.g., Multilayer Perceptron (MLP) [53], Convolutional Neural Network (CNN) [54], and Recurrent Neural Network (RNN) [55]. However, an in-depth discussion is out of the scope of this paper. We refer interested readers to [56]–[61] for in-depth discussions of DNN architectures and training. We next focus on FL, an important paradigm shift towards enabling privacy preserving and collaborative DL model training.

B. Federated Learning

Motivated by privacy concerns among data owners, the concept of FL is introduced in [21]. FL allows users to collaboratively train a shared model while keeping personal data on their devices, thus alleviating their privacy concerns. As such, FL can serve as an enabling technology for ML model training at mobile edge networks.

In general, there are two main entities in the FL system, i.e., the data owners (viz. *participants*) and the model owner (viz. *FL server*). Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of N data owners, each of which has a private dataset $D_{i \in \mathcal{N}}$. Each data owner i uses its dataset D_i to train a *local model* $\mathbf{w}_i$ and send only the local model parameters to the FL server. Then, all collected local models are aggregated $\mathbf{w} = \cup_{i \in \mathcal{N}} \mathbf{w}_i$ to generate a *global model* $\mathbf{w}_G$. This is different from the traditional centralized training which uses $\mathbf{D} = \cup_{i \in \mathcal{N}} D_i$ to train a model $\mathbf{w}_T$, i.e., data from each individual source is aggregated and processed centrally.

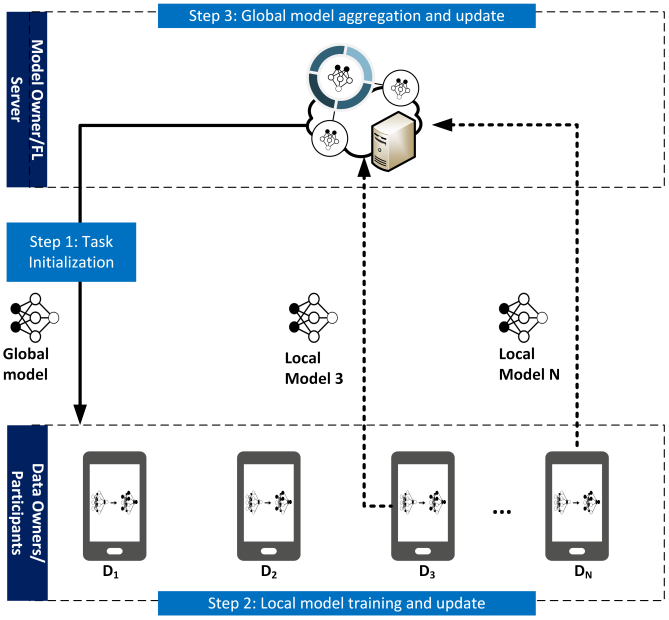

Fig. 4: General FL training process involving N participants.

A typical architecture and training process of an FL system is shown in Fig. 4. In this system, the data owners serve as the FL participants which collaboratively train an ML model required by an aggregate server. An underlying assumption is that the data owners are honest, which means they use their real private data to do the training and submit the true local models to the FL server. Of course, this assumption may not always be realistic [62] and we discuss the proposed solutions subsequently in Sections IV and V.

In general, the FL training process includes the following three steps. Note: the *local* model refers to the model trained at each participating device, whereas the *global* model refers to the model aggregated by the FL server.

- *Step 1 (Task initialization)*: The server decides the training task, i.e., the target application, and the corresponding data requirements. The server also specifies the hyperparameters of the global model and the training process, e.g., learning rate. Then, the server broadcasts the initialized global model $\mathbf{w}_G^0$ and task to selected participants.
- *Step 2 (Local model training and update)*: Based on the global model $\mathbf{w}_G^t$, where t denotes the current iteration index, each participant respectively uses its local data and device to update the local model parameters $\mathbf{w}_i^t$. The goal of participant i in iteration t is to find optimal parameters $\mathbf{w}_i^t$ that minimize the loss function $L(\mathbf{w}_i^t)$, i.e.,
$$\mathbf{w}_i^{t^*} = \arg\min_{\mathbf{w}_i^t} L(\mathbf{w}_i^t). \tag{3}$$
The updated local model parameters are subsequently sent to the server.
- *Step 3 (Global model aggregation and update)*: The server aggregates the local models from participants and then sends the updated global model parameters $\mathbf{w}_G^{t+1}$ back to the data owners.

The server wants to minimize the global loss function $L(\mathbf{w}_G^t)$, i.e.,
$$L(\mathbf{w}_G^t) = \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{w}_i^t). \tag{4}$$
Steps 2-3 are repeated until the global loss function converges or a desirable training accuracy is achieved.

Note that the FL training process can be used for different ML models that essentially use the SGD method such as Support Vector Machines (SVMs) [63], neural networks, and linear regression [64]. A training dataset usually contains a set of n data feature vectors $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a set of corresponding data labels[1] $\mathbf{y} = \{y_1, \ldots, y_n\}$. In addition, let $\hat{y_j} = f(\mathbf{x}_j; \mathbf{w})$ denote the predicted result from the model $\mathbf{w}$ updated/trained by data vector x_j. Table II summarizes several loss functions of common ML models [65].

Global model aggregation is an integral part of FL. A straightforward and classical algorithm for aggregating the local models is the *FedAvg* algorithm proposed in [23], which is based on SGD given in Algorithm 1. As described in Step 1 above, the server first initializes the task (lines 11-16). Thereafter, in Step 2, the participant i implements the local

[1] In the case of unsupervised learning, there is no data label.

TABLE II: Loss functions of common ML models

Model	Loss function $L(\mathbf{w}_i^t)$
Neural network	$\frac{1}{n}\sum_{j=1}^{n}(y_i - f(\mathbf{x}_j;\mathbf{w}))^2$ (Mean Squared Error)
Linear regression	$\frac{1}{2}\left\|y_j - \mathbf{w}^T\mathbf{x}_j\right\|^2$
K-means	$\sum_j \left\|\mathbf{x}_j - f(\mathbf{x}_j;\mathbf{w})\right\|$ ($f(\mathbf{x}_j;\mathbf{w})$ is the centroid of all objects assigned to x_j's class)
squared-SVM	$[\frac{1}{n}\sum_{j=1}^{n}\max(0, 1-y_j(\mathbf{w}^T\mathbf{x}_j - bias))]$ $+\lambda\left\|\mathbf{w}^T\right\|^2$($bias$ is the bias parameter and λ is const.)

Algorithm 1 Federated averaging algorithm [23]

Require: Local minibatch size B, number of participants m per iteration, number of local epochs E, and learning rate η.
Ensure: Global model $\mathbf{w}_G$.
1: [Participant i]
2: **LocalTraining**(i, $\mathbf{w}$):
3: Split local dataset D_i to minibatches of size B which are included into the set $\mathcal{B}_i$.
4: **for** each local epoch j from 1 to E **do**
5: **for** each $b \in \mathcal{B}_i$ **do**
6: $\mathbf{w} \leftarrow \mathbf{w} - \eta\Delta L(\mathbf{w}; b)$ (η is the learning rate and ΔL is the gradient of L on b.)
7: **end for**
8: **end for**
9:
10: [Server]
11: Initialize $\mathbf{w}_G^0$
12: **for** each iteration t from 1 to T **do**
13: Randomly choose a subset $\mathcal{S}_t$ of m participants from $\mathcal{N}$
14: **for** each partipant $i \in \mathcal{S}_t$ **parallely do**
15: $\mathbf{w}_i^{t+1} \leftarrow$ **LocalTraining**(i, $\mathbf{w}_G^t$)
16: **end for**
17: $\mathbf{w}_G^t = \frac{1}{\sum_{i\in\mathcal{N}} D_i}\sum_{i=1}^{N} D_i\mathbf{w}_i^t$ (Averaging aggregation)
18: **end for**

training and optimizes the target in (3) on minibatches from the original local dataset (lines 2-8). Note that a minibatch refers to a randomized subset of each participant's dataset. At the t^{th} iteration (line 17), the server minimizes the global loss in (4) by the averaging aggregation which is formally defined as

$$\mathbf{w}_G^t = \frac{1}{\sum_{i\in\mathcal{N}} D_i}\sum_{i=1}^{N} D_i\mathbf{w}_i^t. \tag{5}$$

The FL training process is iterated till the global loss function converges, or a desirable accuracy is achieved.

C. Statistical Challenges of FL

Following an elaboration of the FL training process in the previous section, we now proceed to discuss the statistical challenges faced in FL.

In traditional distributed ML, the central server has access to the whole training dataset. As such, the server can split the dataset into subsets that follow similar distributions. The subsets are subsequently sent to participating nodes for distributed training. However, this approach is impractical for FL since the local dataset is only accessible by the data owner.

In the FL setting, the participants may have local datasets that follow different distributions, i.e., the datasets of participants are non-IID. While the authors in [23] show that the aforementioned *FedAvg* algorithm is able to achieve desirable accuracy even when data is non-IID across participants, the authors in [66] found otherwise. For example, the accuracy of a *FedAvg*-trained CNN model has 51% lower accuracy than centrally-trained CNN model for CIFAR-10 [67]. This deterioration in accuracy is further shown to be quantified by the earth mover's distance (EMD) [68], i.e., difference in FL participant's data distribution as compared to the population distribution. As such, when data is non-IID and highly skewed, data-sharing is proposed in which a shared dataset with uniform distribution across all classes is sent by the FL server to each FL participant. Then, the participant trains its local model on its private data together with the received data. The simulation result shows that accuracy can be increased by 30% with 5% shared data due to reduced EMD. However, a common dataset may not always be available for sharing by the FL server. An alternative solution is subsequently discussed in section IV.

The authors in [69] also find that global imbalance, i.e., the situation in which the collection of data held across all FL participants is class imbalanced, also leads to a deterioration in model accuracy. As such, the Astraea framework is proposed. On initialization, the FL participants first send their data distribution to the FL server. A rebalancing step is introduced before training begins in which each participant performs data augmentation [70] on the minority classes, e.g., through random rotations and shifts. After training on the augmented data, a mediator is created to coordinate intermediate aggregation, i.e., before sending the updated parameters to the FL server for global aggregation. The mediator selects participants with data distributions that best contributes to an uniform distribution when aggregated. This is done through a greedy algorithm approach to minimize the Kullback-Leibler Divergence [71] between local data and uniform distribution. The simuation results show accuracy improvement when tested on imbalanced datasets.

The data on each participant's device can also be heterogeneous in other ways, e.g., the number of training data owned across participants can differ. The authors in [72] propose learning separate, but structurally related models for each participant. As such, concepts in multi-task learning [73] can naturally be adopted to model such relationships. Instead of minimizing the conventional loss function presented previously in Table II, the loss function is modified to also model the relationship amongst tasks. Then, the MOCHA algorithm is proposed in which an alternating optimization approach [74] is used to approximately solve the minimization problem. Interestingly, MOCHA can be calibrated based on the resource constraints of a participating device. For example, the quality of approximation can be adaptively adjusted based on network conditions and CPU states of the participating devices. However, MOCHA cannot be applied to non-convex DL models.

Apart from data heterogeneity, the convergence of a distributed learning algorithm is always a concern. Higher convergence rate helps to save a large amount of time and resources for the FL participants, and also significantly increases the success rate of the federated training since fewer communication rounds will reduce participant dropouts. To ensure convergence, the study in [75] propose *FedProx*, which modifies the loss function to also include a tunable parameter

that restricts how much local updates can affect the initial model parameters. The *FedProx* algorithm can be adaptively tuned, e.g., when training loss is increasing, model updates can be tuned to affect the current parameters less. Similarly, the authors in [76] also propose the *LoAdaBoost FedAvg* algorithm to complement the aforementioned data-sharing approach [66] in ML on medical data. In *LoAdaBoost FedAvg*, participants train the model on their local data and compare the cross-entropy loss with the median loss from the *previous* training round. If the current cross-entropy loss is higher, the model is retrained before global aggregation so as to increase learning efficiency. The simulation results show that faster convergence is achieved as a result.

In fact, the statistical challenges of FL coexist with other issues that we explore in subsequent sections. For example, the communication costs incurred in FL can be reduced by faster convergence. Similarly, resource allocation policies can also be designed towards solving statistical heterogeneity. As such, we revisit these concepts in greater detail subsequently.

D. FL protocols and frameworks

To improve scalability, an FL protocol has been proposed in [77] from the system level. This protocol deals with issues regarding unstable device connectivity and communication security etc.

The FL protocol (Fig. 5) consists of three phases in each training round:

1) *Selection:* In the participant selection phase, the FL server chooses a subset of connected devices to participate in a training round. The selection criteria may subsequently be calibrated to the server's needs, e.g., training efficiency [78]. In Section IV, we further elaborate on proposed participant selection methods.
2) *Configuration:* The server is configured accordingly to the aggregation mechanism preferred, e.g. simple or secure aggregation [79]. Then, the server sends the training schedule and global model to each participant.
3) *Reporting:* The server receives updates from participants. Thereafter, the updates can be aggregated, e.g., using the *FedAvg* algorithm.

In addition, to manage device connections accordingly to varying FL population size, pace steering is also recommended. Pace steering adaptively manages the optimal time window for participants to reconnect to the FL server [77]. For example, when the FL population is small, pace steering is used to ensure that there is a sufficient number of participating devices that connect to the server simultaneously. In contrast, when there is a large population, pace steering randomly chooses devices to participate to prevent the situation in which too many participating devices are connected at one point of time.

Apart from communication efficiency, communication security during local updates transmission is another problem to be resolved. Specifically, there are mainly two aspects in communication security:

1) *Secure aggregation*: To prevent local updates from being traced and utilized to infer the identity of the FL participant, a virtual and trusted third party server is deployed for local model aggregation [79]. The Secret Sharing mechanism [80] is also used for transmission of local updates with authenticated encryption.
2) *Differential privacy*: Similar to secure aggregation, differential privacy (DP) prevents the FL server from identifying the owner of a local update. The difference is that to achieve the goal of privacy preservation, the DP in FL [81] adds a certain degree of noise in the original local update while providing theoretical guarantees on the model quality.

These concepts on privacy and security are presented in detail in Section V. Recently, some open-source frameworks for FL have been developed as follows:

1) *TensorFlow Federated (TFF)*: TFF [82] is a framework based on Tensorflow developed by Google for decentralized ML and other distributed computations. TFF consists of two layers (i) FL and (ii) Federated Core (FC). The FL layer is a high-level interface that allows the implementation of FL to existing TF models without the user having to apply the FL algorithms personally. The FC layer combines TF with communication operators to allow users to experiment with customized and newly designed FL algorithms.
2) *PySyft*: PySyft [83] is a framework based on PyTorch for performing encrypted, privacy-preserving DL and implementations of related techniques, such as Secure Multiparty Computation (SMPC) and DP, in untrusted environments while protecting data. Pysyft is developed such that it retains the native Torch interface, i.e., the ways to execute all tensor operations remain unchanged from that of Pytorch. When a SyftTensor is created, a LocalTensor is automatically created to also apply the input command to the native Pytorch tensor. To simulate FL, participants are created as *Virtual Workers*. Data, i.e., in the structure of tensors, can be split and distributed to the Virtual Workers as a simulation of a practical FL setting. Then, a PointerTensor is created to specify the data owner and storage location. In addition, model updates can be fetched from the Virtual Workers for global aggregation.
3) *LEAF*: An open source framework [84] of datasets that can be used as benchmarks in FL, e.g., Federated Extended MNIST (FEMNIST), an MNIST [85] dataset partitioned based on writer of each character, and Sentiment140 [86], a dataset partitioned based on different users. In these datasets, the writer or user is assumed to be a participant in FL, and their corresponding data is taken to be the local data held in their personal devices. The implementation of newly designed algorithms on these benchmark datasets allow for reliable comparison across studies.

E. Unique characteristics and issues of FL

Besides the statistical challenges we present in Section II-C, FL has some unique characteristics and features [87] as compared to other distributed ML approaches:

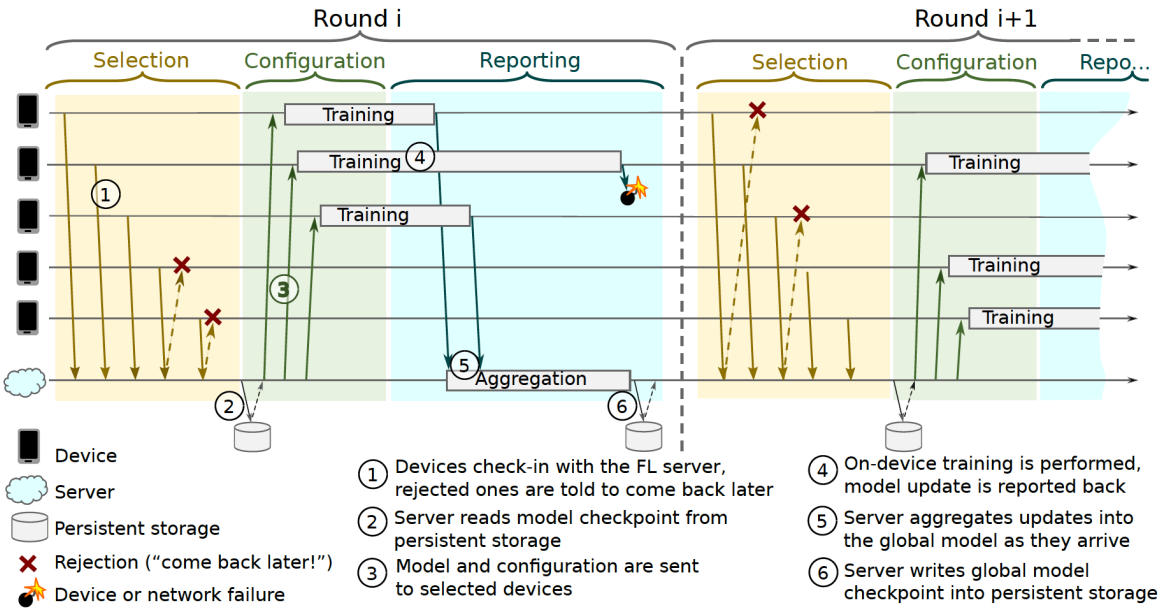

Fig. 5: Federated learning protocol [77].

1) *Slow and unstable communication*: In the traditional distributed training in a data center, the communication environment can be assumed to be perfect where the information transmission rate is very high and there is no packet loss. However, these assumptions are not applicable to the FL environment where heterogeneous devices are involved in training. For example, the Internet upload speed is typically much slower than download speed [88]. Also, some participants with unstable wireless communication channels may consequently drop out due to disconnection from the Internet.
2) *Heterogeneous devices*: Apart from bandwidth constraints, FL involves heterogeneous devices with varying resource constraints. For example, the devices can have different computing capabilities, i.e., CPU states and battery level. The devices can also have different levels of *willingness* to participate, i.e., FL training is resource consuming and given the distributed nature of training across numerous devices, there is a possibility of free ridership.
3) *Privacy and security concerns*: As we have previously discussed, data owners are increasingly privacy sensitive. However, as will be subsequently presented in Section V, malicious participants are able to infer sensitive information from shared parameters, which potentially negates privacy preservation. In addition, we have previously assumed that all participants and FL servers are trustful. In reality, they may be malicious.

These unique characteristics of FL lead to several practical issues in FL implementation mainly in three aspects that we now proceed to discuss, i.e., (i) communication costs (ii) resource allocation and (iii) privacy and security. In the following sections, we review related work that address each of these issues.

III. Communication Cost

In FL, a number of rounds of communications between the participants and the FL server may be required to achieve a target accuracy (Fig. 5). For complex DL model training involving, e.g. CNN, each update may comprise millions of parameters [89]. The high dimensionality of the updates can result in the incurrence of high communication costs and can lead to a training bottleneck. In addition, the bottleneck can be worsened due to (i) unreliable network conditions of participating devices [90] and (ii) the asymmetry in Internet connection speeds in which upload speed is faster than download speed, resulting in delays in model uploads by participants [88]. As such, there is a need to improve the communication efficiency of FL. The following approaches to reduce communication costs are considered:

- *Edge and End Computation*: In the FL setup, the communication cost often dominates computation cost [23]. The reason is that on-device dataset is relatively small whereas mobile devices of participants have increasingly fast processors. On the other hand, participants may only be willing to participate in model training only if they are connected to Wi-Fi [88]. As such, more computation can be performed on edge nodes or on end devices before each global aggregation so as to reduce the number of communication rounds needed for model training. In addition, means to ensure faster convergence can also reduce number of communication rounds involved, at the expense of more computation on edge servers and end devices.
- *Model Compression*: This is a technique commonly used in distributed learning [91]. Model compression involves the communication of a model update that is transformed to be more compact, e.g., through sparsification, quantization or subsampling [92]. However, since the compression may introduce noise, the objective is to reduce the size of

update transferred during each round of communication while maintaining the quality of trained models [93].

- *Importance-based Updating*: This strategy involves selective communication such that only the important or relevant updates [94] are transmitted in each communication round.

A. Edge and End Computation

To decrease the number of communication rounds, additional computation can be performed on participating end devices before each iteration of global aggregation (Fig. 6(a)). The authors in [23] consider two ways to increase computation on participating devices: (i) increasing parallelism in which more participants are selected to participate in each round of training and (ii) increasing computation per participant whereby each participant performs more local updates before communication for global aggregation. A comparison is conducted for the FederatedSGD (*FedSGD*) algorithm and the proposed *FedAvg* algorithm. For the *FedSGD* algorithm, all participants are involved and only one pass is made per training round in which the minibatch size comprises of the entirety of the participant's dataset. This is similar to the full-batch training in centralized DL frameworks. For the proposed *FedAvg* algorithm, the hyperparameters are tuned such that more local computation is performed by the participants. For example, the participant can make more passes over its dataset or use a smaller local minibatch size to increase computation before each communication round. The simulation results show that increased parallelism does not lead to significant improvements in reducing communication cost, once a certain threshold is reached. As such, more emphasis is placed on increasing computation per participant while keeping the fraction of selected participants constant. For MNIST CNN simulations, increased computation using the proposed *FedAvg* algorithm can reduce communication rounds by more than 30 times when the dataset is IID. For non-IID dataset, the improvement is less significant (2.8 times) using the same hyperparameters. However, for Long Short Term Memory (LSTM) simulations [95], improvements are more significant even for non-IID data (95.3 times). In addition, *FedAvg* increases the accuracy eventually since model averaging produces regularization effects similar to dropout [96], which prevents overfitting.

One way to decrease communication cost can also be through modifying the training algorithm to increase convergence speed, e.g., through the aforementioned *LoAdaBoost FedAvg* in [76]. Similarly, the authors in [97] also propose increased computation on each participating device by adopting a two-stream model (Fig 6(b)) commonly used in transfer learning and domain adaption [99]. During each training round, the global model is received by the participant and fixed as a reference in the training process. During training, the participant learns not just from local data, but also from other participants with reference to the fixed global model. This is done through the incorporation of Maximum Mean Discrepancy (MMD) into the loss function. MMD measures the distance between the means of two data distributions [99],

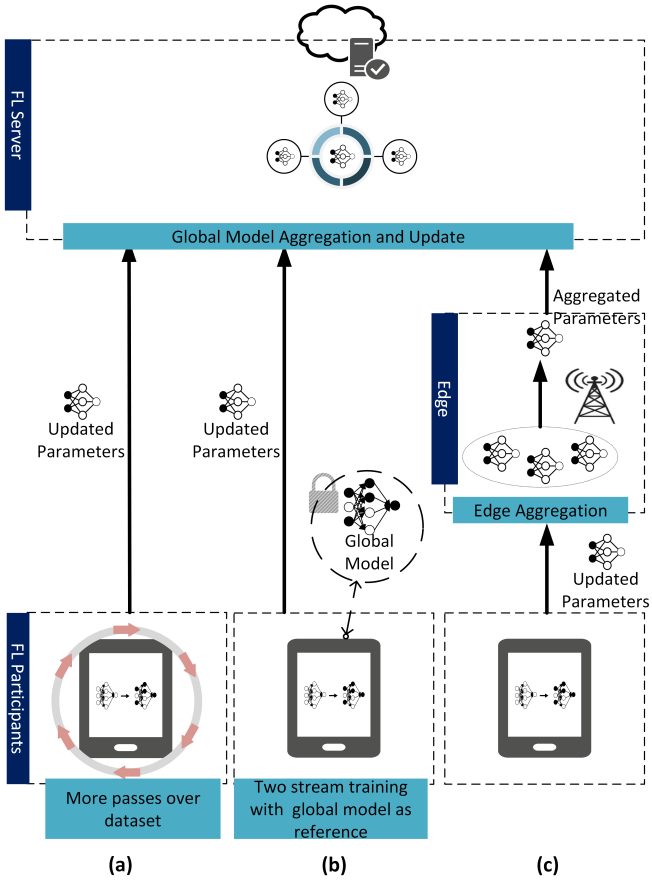

Fig. 6: Approaches to increase computation at edge and end devices include (a) Increased computation at end devices, e.g., more passes over dataset before communication [23] (b) Two-stream training with global model as a reference [97] and (c) Intermediate edge server aggregation [98]

[100]. Through minimizing MMD loss between the local and global models, the participant can extract more generalized features from the global model, thus accelerating the convergence of training process and reducing communication rounds. The simulation results on the CIFAR-10 and MNIST dataset using DL models such as AlexNet [54] and 2-CNN respectively show that the proposed two-stream FL can reach the desirable test accuracy in 20% fewer communication rounds even when data is non-IID. However, while convergence speed is increased, more computation resources have to be consumed by end devices for the aforementioned approaches. As such, this necessitates resource allocation optimization that we subsequently discuss in Section IV.

While the aforementioned studies consider increasing computation on participating *devices*, the authors in [98] propose that proximate edge *servers* can serve as intermediary parameter aggregators given that the propagation latency from participant to the edge server is smaller than that of the participant-server communication (Fig 6(c)). A hierarchical FL (*HierFAVG*) algorithm is proposed whereby for every few local participant updates, the edge server aggregates the collected local models. After a predefined number of edge server

aggregations, the edge server communicates with the cloud for global model aggregation. As such, the communication between the participants and the cloud occurs only once after an interval of multiple local updates. Comparatively, for the *FedAvg* algorithm proposed in [23], the global aggregation occurs more frequently since no intermediate edge server aggregation is involved. The authors further prove the convergence of *HierFAVG* for both convex and non-convex objective functions given non-IID user data. The simulation results show that for the same number of local updates between two global aggregations, more intermediate edge aggregations before each global aggregation can lead to reduced communication overhead as compared to the *FedAvg* algorithm. This result holds for both IID and non-IID data, implying that intermediate aggregation on edge servers may be implemented on top of *FedAvg* so as to reduce communication costs. However, when applied to non-IID data, the simulation results show that *HierFAVG* fails to converge to the desired accuracy level (90%) in some instances, e.g., when edge-cloud divergence is large or when there are many edge servers involved. As such, a further study is required to better understand the tradeoffs between adjusting local and edge aggregation intervals, so as to ensure that the parameters of the *HierFAVG* algorithm can be optimally calibrated to suit other settings. Nevertheless, *HierFAVG* is a promising approach for the implementation of FL at mobile edge networks, since it leverages on the proximity of intermediate edge server to reduce communication costs, and potentially relieve the burden on the remote cloud.

B. Model Compression

To reduce communication costs, the authors in [88] propose structured and sketched updates to reduce the size of model updates sent from participants to the server during each communication round. Structured updates restrict participant updates to have a pre-specified structure, i.e., low rank and random mask. For the low rank structure, each update is enforced to be a low rank matrix expressed as a product of two matrices. Here, one matrix is generated randomly and held constant during each communication round whereas the other is optimized. As such, only the optimized matrix needs to be sent to the server. For the random mask structure, each participant update is restricted to be a sparse matrix following a pre-defined random sparsity pattern generated independently during each round. As such, only the non-zero entries are required to be sent to the server.

On the other hand, sketched updates refer to the approach of encoding the update in a compressed form before communication with the server, which subsequently decodes the updates before aggregation. One example of sketched update is the subsampling approach, in which each participant communicates only a random subset of the update matrix. The server then averages the subsampled updates to derive an unbiased estimate of the true average. Another example of sketched update is the probabilistic quantization approach [101], in which the update matrices are vectorized and quantized for each scalar. To reduce the error from quantization, a structured random rotation that is the product of a Walsh-Hadamard matrix and binary diagonal matrix [102] can be applied before quantization. The simulation results on the CIFAR-10 image classification task show that for structured updates, the random mask performs better than that of the low rank approach. The random mask approach also achieves higher accuracy than sketching approaches since the latter involves a removal of some information obtained during training. However, the combination of all three sketching tools, i.e., subsampling, quantization, and rotation, can achieve higher compression rate and faster convergence, albeit with some sacrifices in accuracy. For example, by using 2 bits for quantization and sketching out all but 6.25% of update data, the number of bits needed to represent updates can be reduced by 256 times and the accuracy level achieved is 85%. In addition, sketching updates can achieve higher accuracy in training when there are more participants trained per round. This suggests that for practical implementation of FL where there are many participants available, more participants can be selected for training per round so that subsampling can be more aggressive to reduce communication costs.

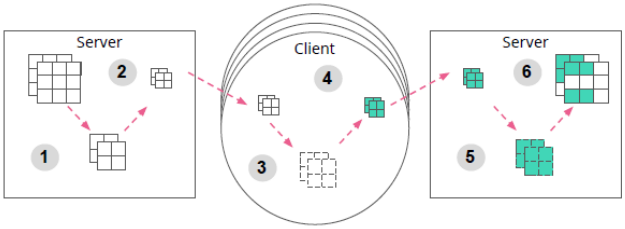

Fig. 7: The compression techniques considered are summarized above by the diagram from authors in [93]. (i) Federated dropout to reduce size of model (ii) Lossy compression of model (iii) Decompression for training (iv) Compression of participant updates (v) Decompression (vi) Global aggregation

The authors in [93] extend on the studies in [88] by proposing lossy compression and federated dropout to reduce *server-to-participant* communication costs. A summary of the proposed techniques are adapted from the authors' work in Fig. 7. For participant-to-server communication of model parameters that we discuss previously, the decompressions can be averaged over many updates to receive an unbiased estimate. However, there is no averaging for server-to-participant communications since the same global model is sent to all participants during each round of communication. Similar to [88], subsampling and probabilistic quantization are considered. For the application of structured random rotation before subsampling and quantization, Kashin's representation [103] is applied instead of the Hadamard transformation since the former is found to perform better in terms of accuracy-size trade-off.

In addition to the subsampling and quantization approaches, the federated dropout approach is also considered in which a fixed number of activation functions at each fully-connected layer is removed to derive a smaller sub-model. The sub-model is then sent to the participants for training. The updated submodel can then be mapped back to the global model to derive a complete DNN model with all weights updated during subsequent aggregation. This approach reduces the

server-to-participant communication cost, and also the size of participant-to-server updates. In addition, local computation is reduced since fewer parameters have to be updated. The simulations are performed on MNIST, CIFAR-10 and EMNIST [104] datasets. For the lossy compression, it is shown that the subsampling approach taken by [88] does not reach an acceptable level of performance. The reason is that the update errors can be averaged out for participant-to-server uploads but not for server-to-participant downloads. On the other hand, quantization with Kashin's representation can achieve the same performance as the baseline without compression while having communication cost reduced by nearly 8 times when the model is quantized to 4 bits. For federated dropout approaches, the results show that a dropout rate of 25% of weight matrices of fully-connected layers (or filters in the case of CNN) can achieve acceptable accuracy in most cases while ensuring around 43% reduction in size of models communicated. However, if dropout rates are more aggressive, convergence of the model can be slower.

The aforementioned two studies suggest useful model compression approaches in reducing communication costs for both server-to-participant and participant-to-server communications. As one may expect, the reduction in communication costs come with sacrifices in model accuracy. It will thus be useful to formalize the compression-accuracy tradeoffs, especially since this varies for different tasks, or when different number of FL participants are involved.

C. Importance-based Updating

Based on the observation that most weight values of a DNN are sparsely distributed and close to zero [105], the authors in [94] propose the edge Stochastic Gradient Descent (eSGD) algorithm that selects only a small fraction of important gradients to be communicated to the FL server for parameter update during each communication round. The eSGD algorithm keeps track of loss values at two consecutive training iterations. If the loss value of the current iteration is smaller than the preceding iteration, this implies that current training gradients and model parameters are important for training loss minimalization and thus, their respective hidden weights are assigned a positive value. In addition, the gradient is also communicated to the server for parameter update. Once this does not hold, i.e., the loss increases as compared to the previous iteration, other parameters are selected to be updated based on their hidden weight value. A parameter with larger hidden weight value is more likely to be selected since it has been labeled as important several times during training. To account for small gradient values that can delay convergence if they are ignored and not updated completely [106], these gradient values are accumulated as residual values. Since the residuals may arise from different training iterations, each update to the residual is weighted with a discount factor using the momentum correction technique [107]. Once the accumulated residual gradient reaches a threshold, they are chosen to replace the least important gradient coordinates according to the hidden weight values. The simulation results show that eSGD with a 50% drop ratio can achieve higher accuracy than that of the thresholdSGD algorithm proposed in [105], which uses a fixed threshold value to determine which gradient coordinates to drop. eSGD can also save a large proportion of gradient size communicated. However, eSGD still suffers from accuracy loss as compared to standard SGD approaches. For example, when tested on simple classification tasks using the MNIST dataset, the model accuracy converges to just 91.22% whereas standard SGD can achieve 99.77% accuracy. If extended to more sophisticated tasks, the accuracy can potentially deteriorate to a larger extent. In addition, the accuracy and convergence speed of the eSGD approach fluctuates arbitrarily based on hyperparameters used, e.g., minibatch size. As such, further studies have to be conducted to formally balance the tradeoffs between communication costs and training performance.

Similar to [94], the authors in [90] propose the Communication-Mitigated Federated Learning (CMFL) algorithm that uploads only relevant local updates to reduce communication costs while guaranteeing global convergence. In each iteration, a participant's local update is first compared with the global update to identify if the update is relevant. A relevance score is computed where the score equates to percentage of same-sign parameters in the local and global update. In fact, the global update is not known in advance before aggregation. As such, the global update made in the previous iteration is used as an estimate for comparison since it was found empirically that more than 99% of the normalized difference of two sequential global updates are smaller than 0.05 in both MNIST CNN and Next-Word-Prediction LSTM. An update is considered to be irrelevant if its relevance score is smaller than a predefined threshold. The simulation results show that CMFL requires 3.47 times and 13.97 times fewer communication rounds to reach 80% accuracy for MNIST CNN and Next-Word-Prediction LSTM, respectively, as compared to the benchmark *FedAvg* algorithm. In addition, CMFL can save significantly more communication rounds as compared to Gaia. Note that Gaia is a geo-distributed ML approach suggested in [108] which measures relevance based on *magnitude* of updates rather than sign of parameters. When applied with the aforementioned MOCHA algorithm II-C [72], CMFL can reduce communication rounds by 5.7 times for the Human Activity Recognition dataset [109] and 3.3 times for the Semeion Handwritten Digit dataset [110]. In addition, CMFL can achieve slightly higher accuracy since it involves the elimination of irrelevant updates that are outliers which harm training.

Summary: In this section, we have reviewed three main approaches to communication cost reduction in FL, and for each approach, we discuss the solutions proposed in different studies. We summarize the approaches along with references in Table III. Communication cost is a key issue to be solved before we can implement FL at scale. In our previous discussion, we note that many of the approaches to reduce communication costs come with sacrifices in other aspects, e.g., deterioration in model accuracy and increased computation on end devices. As such, for successful communication cost reduction, this tradeoff has to be well managed. In addition, the participating devices also have other resource constraints that can affect

TABLE III: Approaches to communication cost reduction in FL.

Approaches	Ref.	Key Ideas
Edge and End Computation	[23]	More local updates before communication
	[97]	Reference to global model for faster convergence
	[98]	Intermediate edge aggregation before FL server aggregation
Model Compression	[88]	Structured and sketched updates for participant-to-server communication
	[93]	Lossy compression and federated dropout for server-to-participant communication
Importance-based Updating	[94]	eSGD to selectively communicate parameters that reduce training loss
	[90]	CMFL to selectively communicate parameters based on signs of parameters compared to global parameters

training efficiency. In particular, FL involves heterogeneous and distributed devices with varying resource constraints. As such, we now proceed to review issues in resource allocation in the following section.

IV. Resource Allocation

FL involves the participation of heterogeneous devices that have different dataset qualities, computation capabilities, energy states, and willingness to participate. Given the device heterogeneity and resource constraints, i.e., in device energy states and communication bandwidth, resource allocation has to be optimized to maximize the efficiency of the training process. In particular, the following resource allocation issues need to be considered:

- *Participant Selection:* As part of the FL protocol presented in Section II-D, participant selection refers to the selection of devices to participate in each training round. Typically, a set of participants is randomly selected by the server to participate. Then, the server has to aggregate parameter updates from all participating devices in the round before taking a weighted average of the models [23]. As such, the training progress of FL is limited by the training time of the slowest participating devices, i.e., stragglers [111]. This results in the training bottleneck. New participant selection protocols are investigated in order to address the training bottleneck in FL.
- *Adaptive Aggregation:* As discussed in Section II-B, FL involves global aggregation in which model parameters are communicated to the FL server for aggregation. The conventional approach to global aggregation is a synchronous one, i.e., in which aggregations occur in fixed intervals. However, adaptive calibrations of global aggregation frequency are investigated to increase training efficiency subject to resource constraints [111].
- *Incentive Mechanism:* In the practical implementation of FL, participants may be reluctant to participate in a federation without receiving compensation since training models is resource-consuming. In addition, there exists information asymmetry between the FL server and participants since participants have greater knowledge of their available computation resources and data quality. Therefore, incentive mechanisms have to be carefully designed to both incentivize participation and reduce the potential adverse impacts of information asymmetry.

A. Participant Selection

To mitigate the training bottleneck, the authors in [78] propose a new FL protocol called *FedCS*. This protocol is illustrated in Fig. 8. The system model is a MEC framework in which the operator of the MEC is the FL server that coordinates training in a cellular network that comprises participating mobile devices that have heterogeneous resources. Accordingly, the FL server first conducts a *Resource Request* step to gather information such as wireless channel states and computing capabilities from a subset of randomly selected participants. Based on this information, the MEC operator selects the maximum possible number of participants that can complete the training within a prespecified deadline for the subsequent global aggregation phase. By selecting the maximum possible number of participants in each round, accuracy and efficiency of training are preserved. To solve the maximization problem, a greedy algorithm [112] is proposed, i.e., participants that take the least time for model upload and update are iteratively selected for training. The simulation results show that compared with the FL protocol which only accounts for training deadline without performing participant selection, *FedCS* can achieve higher accuracy since *FedCS* is able to involve more participants in each training round [23]. However, *FedCS* has been tested only on simple DNN models. When extended to the training of more complex models, it may be difficult to estimate how many participants should be selected. For example, more training rounds may be needed for the training of complex models, and the selection of too few participants may lead to poor performance considering that some participants may drop out during training. In addition, there is bias towards selecting participants with devices that have better computing capabilities. These participants may not hold data that is representative of the population distribution. In particular, we revisit the fairness issue [113] subsequently in this section.

While *FedCS* addresses heterogeneity of resources among participants in FL, the authors in [114] extend their work on the *FedCS* protocol with the Hybrid-FL protocol that deals with differences in data distributions among participants. The dataset of participants participating in FL may be non-IID since it is reflective of each individual user's specific characteristics. As we have discussed in Section II-C, the non-IID dataset may significantly degrade the performance of the *FedAvg* algorithm [66]. One proposed measure to address the non-IID nature of the dataset is to distribute publicly available data to participants, such that the EMD between their on-device dataset and the population distance is reduced. However, such a dataset may not always exist, and participants may not download them for security reasons. Thus, an alternative solution is to construct an approximately IID dataset

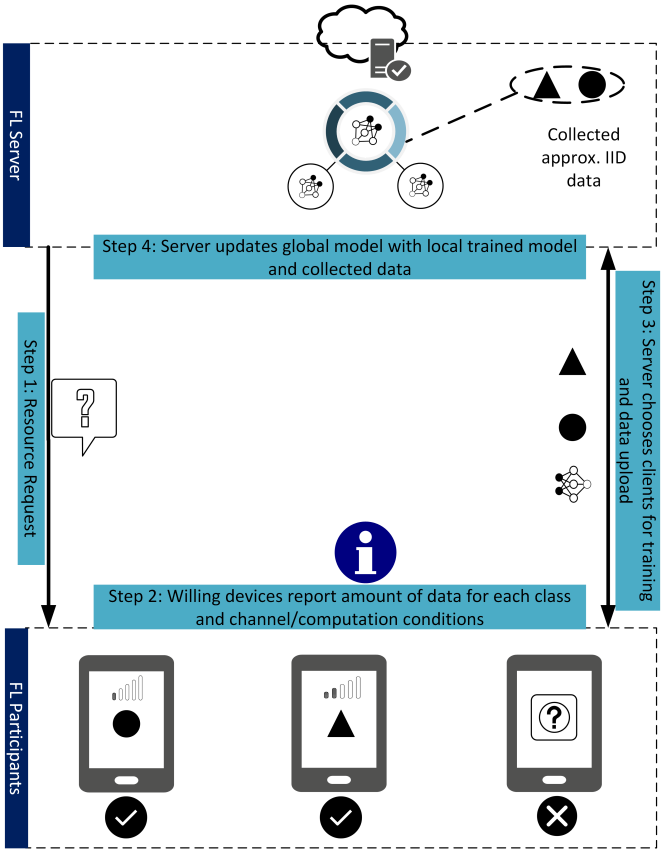

Fig. 8: Participant selection under the FedCS and Hybrid-FL protocol.

using inputs from a limited number of privacy insensitive participants [114]. In the Hybrid-FL protocol, during the *Resource Request* step (Fig. 8), the MEC operator asks random participants if they permit their data to be uploaded. During the participant selection phase, apart from selecting participants based on computing capabilities, participants are selected such that their uploaded data can form an approximately IID dataset in the server, i.e., the amount of collected data in each class has close values (Fig. 8). Thereafter, the server trains a model on the collected IID dataset and merge this model with the global model trained by participants. The simulation results show that even with just 1% of participants sharing their data, classification accuracy for non-IID data can be significantly improved as compared to the aforementioned *FedCS* benchmark where data is not uploaded at all. However, the recommended protocol can violate the privacy and security of users, especially if the FL server is malicious. In the case when participants are malicious, data can be falsified before uploading, as we will further discuss in Section V. In addition, the proposed measure can be costly especially in the case of videos and images. As such, it is unlikely that participants will volunteer for data uploading when they can free ride on the efforts of other volunteers. For feasibility, a well-designed incentive and reputation mechanism is needed to ensure that only trustworthy participants are allowed to upload their data.

In general, the mobile edge network environment in which FL is implemented on is dynamic and uncertain with variable constraints, e.g., wireless network and energy conditions. Thus, this can lead to training bottlenecks. To this end, Deep Q-Learning (DQL) can be used to optimize resource allocation for model training as proposed in [115]. The system model is a Mobile Crowd Machine Learning (MCML) setting which enables participants in a mobile crowd network to collaboratively train DNN models required by a FL server. The participating mobile devices are constrained by energy, CPU, and wireless bandwidth. Thus, the server needs to determine proper amounts of data, energy, and CPU resources that the mobile devices use for training to minimize energy consumption and training time. Under the uncertainty of the mobile environment, a stochastic optimization problem is formulated. In the problem, the server is the agent, the state space includes the CPU and energy states of the mobile devices, and the action space includes the number of data units and energy units taken from the mobile devices. To achieve the objective, the reward function is defined as a function of the accumulated data, energy consumption, and training latency. To overcome the large state and action space issues of the server, the DQL technique based on Double Deep Q-Network (DDQN) [116] is adopted to solve the server's problem. The simulation results show that the DQL scheme can reduce energy consumption by around 31% compared with the greedy algorithm, and training latency is reduced up to 55% compared with the random scheme. However, the proposed scheme is applicable only in federations with few participating mobile devices. As an extension, the scalability of the DQL approach in large federations may be considered.

The aforementioned resource allocation approaches focus on improving the training efficiency of FL. However, this may result in unfair resource allocation, a topic that is commonly explored in resource allocation for wireless networks [117] and ML [118]. For example, if the participant selection protocol selects mobile devices with higher computing capabilities to participate in each training round [78], the FL model will be overrepresented by the distribution of data owned by participants with devices that have higher computing capabilities. Therefore, the authors in [113] and [119] consider fairness as an additional objective in FL. Fairness is defined in [119] to be the *variance* of performance of an FL model across participants. If the variance of the testing accuracy is large, this implies the presence of more bias or less fairness, since the learned model may be highly accurate for certain participants and less so for other underrepresented participants. The authors in [119] propose the q-Fair FL (q-FFL) algorithm that reweighs the objective function in *FedAvg* to assign higher weights in the loss function to devices with higher loss. In fact, this is a generalization of the Agnostic FL (AFL) algorithm proposed in [113], in which the device with the highest loss dominates the entire loss function. The simulation results show that the proposed q-FFL can achieve lower variance of testing accuracy and converges more quickly than the AFL algorithm. However, as expected, for some calibrations of the q-FFL algorithm, there can be convergence slowdown since stragglers can delay the training process. As such, an asynchronous aggregation approach (to be subsequently discussed in this section) can potentially be considered for use with the q-FFL algorithm.

While most of the existing studies have considered FL using orthogonal-access schemes such as Orthogonal Frequency-division Multiple Access (OFDMA) [120], the authors in [121] propose a multi-access Broadband Analog Aggregation (BAA) design for communication-latency reduction in FL. Instead of performing communication and computation separately during global aggregation at the server, the BAA scheme builds on the concept of over-the-air computation [122] to *integrate* computation and communication through exploiting the signal superposition property of a multiple-access channel. The proposed BAA scheme allows the reuse of the whole bandwidth (Fig. 9(a)) whereas OFDMA orthogonalizes bandwidth allocation (Fig. 9(b)). As such, for orthogonal-access schemes, communication latency increases in direct proportion with the number of participants whereas for multi-access schemes, latency is independent of the number of participants. The bottleneck of signal-to-noise ratio (SNR) during BAA transmission is the participating device with the longest propagation distance given that devices that are nearer have to lower their transmission power for amplitude alignment with devices located further. To increase SNR, participants with longer propagation distance have to be dropped. However, this leads to the truncation of model parameters. As such, to manage the SNR-truncation tradeoff, three scheduling schemes are considered namely i) *Cell-interior scheduling*: participants beyond a distance threshold are not scheduled, ii) *All-inclusive scheduling*: all participants are considered, and iii) *Alternating scheduling*: edge server alternates between the two aforementioned schemes. The simulation results show that the proposed BAA scheme can achieve similar test accuracy as the OFDMA scheme while achieving latency reduction from 10 times to 1000 times. As a comparison between the three scheduling schemes, the cell-interior scheme outperforms the all-inclusive scheme in terms of test accuracy for high mobility networks where participants have rapidly changing locations. For low mobility networks, the alternating scheduling scheme outperforms cell-interior scheduling.

As an extension, the authors in [123] also introduce error accumulation and gradient sparsification in addition to over-the-air computation. In [121], gradient vectors that are not transmitted as a result of power constraints are completely dropped. To improve the model accuracy, the untransmitted gradient vectors can first be stored in an error accumulation vector. In the next round, local gradient estimates are then corrected using the error vector. In addition, when there are bandwidth limitations, the participating device can apply gradient sparsification to keep only elements with the highest magnitudes for transmission. The elements that are not transmitted are subsequently added on to the error accumulation vector for gradient estimate correction in the next round. The simulation results show that the proposed scheme can achieve higher test accuracy than over-the-air computation without error accumulation or gradient sparsification since it corrects gradient estimates with the error accumulation vector and allows for a more efficient utilization of the bandwidth.

Similar to [121] and [123], the authors in [124] propose an integration of computation and communication via over-the-air computation. However, it is observed that aggregation

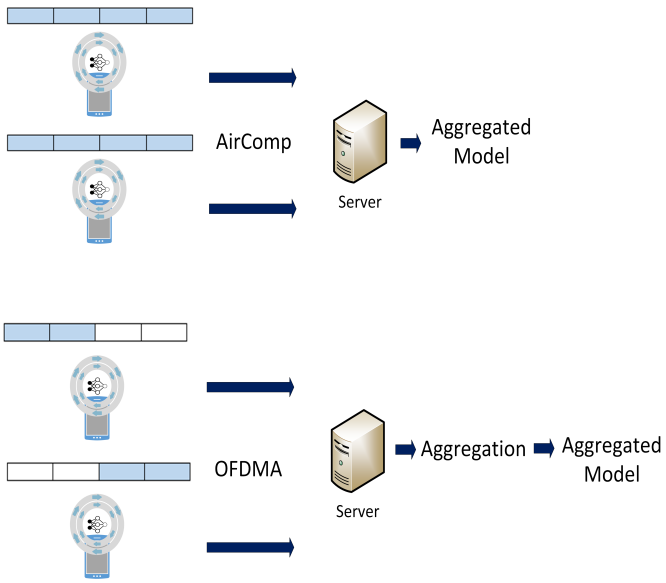

Fig. 9: A comparison [121] between (a) BAA by over-the-air computation which reuses bandwidth (above) and (b) OFDMA (below).

error incurred during over-the-air computation can lead to a drop in model accuracy [125] as a result of signal distortion. As such, a participant selection algorithm is proposed in which the number of devices selected for training is maximized so as to improve statistical learning performance [23] while keeping the signal distortion below a threshold. Due to the nonconvexity [126] of the mean-square-error (MSE) constraint and intractability of the optimization problem, a difference-of-convex functions (DC) algorithm [127] is proposed to solve the maximization problem. The simulation results show that the proposed DC algorithm is scalable and can also achieve near-optimal performance that is comparable to global optimization, which is non-scalable due to its exponential time complexity. In comparison with other state-of-the-art approaches such as the semidefinite relaxation technique (SDR) proposed in [128], the proposed DC algorithm can also select more participants, thus also achieving higher model accuracy.

B. Adaptive Aggregation

The proposed *FedAvg* algorithm synchronously aggregates parameters as shown in Fig. 10(a) and is thus susceptible to the straggler effect, i.e., each training round only progresses as fast as the slowest device since the FL server waits for *all* devices to complete local training before global aggregation can take place [111]. In addition, the model does not account for participants that can join halfway when the training round is already in progress. As such, the asynchronous model is proposed to improve the scalability and efficiency of FL. For asynchronous FL, the server updates the global model whenever it receives a local update (Fig. 10(b)). The authors in [111] find empirically that an asynchronous approach is robust to participants joining halfway during a training round, as well as when the federation

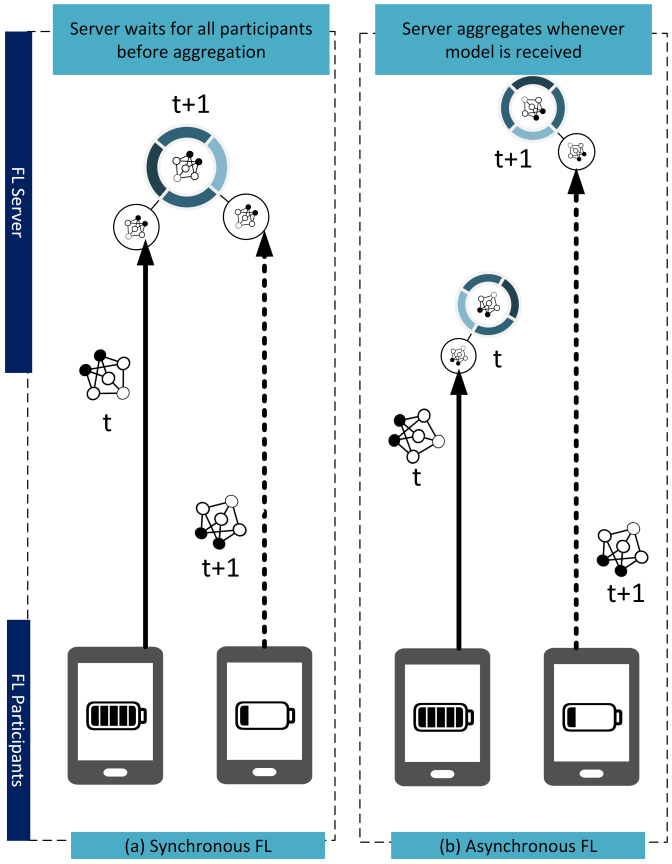

Fig. 10: A comparison between (a) synchronous and (b) asynchronous FL.

involves participating devices with heterogeneous processing capabilities. However, the model convergence is found to be significantly delayed when data is non-IID and unbalanced. As an improvement, [129] propose the *FedAsync* algorithm in which each newly received local updates are adaptively weighted according to staleness, that is defined as the difference between the current epoch and iteration in which the received update belongs to. For example, a stale update from a straggler is outdated since it should have been received in previous training rounds. As such, it is weighted less. In addition, the authors also prove the convergence guarantee for a restricted family of non-convex problems. However, the current hyperparameters of the *FedAsync* algorithm still have to be tuned to ensure convergence in different settings. As such, the algorithm is still unable to generalize to suit the dynamic computation constraints of heterogeneous devices. In fact, given the uncertainty surrounding the reliability of asynchronous FL, synchronous FL remains to be the approach most commonly used today [77].

For most existing implementations of the *FedAvg* algorithm, the global aggregation phase occurs after a fixed number of training rounds. To better manage the dynamic resource constraints, the authors in [65] propose an adaptive global aggregation scheme which varies the global aggregation frequency so as to ensure desirable model performance while ensuring an efficient use of available resources, e.g., energy, during the FL training process. In [65], the MEC system model used consists of (i) the local update phase where the model is trained using local data, (ii) edge aggregation phase where the intermediate aggregation occurs and (iii) global aggregation phase where updated model parameters are received and aggregated by the FL server. In particular, the authors study how the training loss is affected when the total number of edge server aggregation and local updates between global aggregation intervals vary. For this, a convergence bound of gradient descent with non-IID data is first derived. Then, a control algorithm is subsequently proposed to adaptively choose the optimal global aggregation frequency based on the most recent system state. For example, if global aggregation is too time consuming, more edge aggregations will take place before communication with the FL server is initiated. The simulation results show that the adaptive aggregation scheme outperforms the fixed aggregation scheme in terms of loss function minimization and accuracy within the same time budget. However, the convergence guarantee of the adaptive aggregation scheme is only considered for convex loss functions currently.

C. Incentive Mechanism

The authors in [130] propose a service pricing scheme in which participants serve as training service providers for a model owner. In addition, to overcome energy inefficiency in the transfer of model updates, a cooperative relay network is proposed to support model update transfer and trading. The interaction between participants and model owner is modelled as a Stackelberg game [131] in which the model owner is the buyer and participants are the sellers. The Stackelberg game is proposed in which each rational participant can non-cooperatively decide on its own profit maximization price. In the lower-level subgame, the model owner determines size of training data to maximize profits with consideration of the increasing concave relationship between learning accuracy of the model and size of training data. In the upper-level subgame, the participants decide the price per unit of data to maximize their individual profits. The simulation results show that the proposed mechanism can ensure uniqueness of the Stackelberg equilibrium. For example, model updates that contain valuable information are priced higher at the Stackelberg equilibrium. In addition, model updates can be transferred cooperatively, thus reducing congestion in communication and improving energy efficiency. However, the simulation environment only involves relatively few mobile devices.

Similar to [130], the authors in [132] also model the interaction between participants and model owner as a Stackelberg game. However, in this setting, participants are incentivized to allocate more computation power for training. In the lower-level subgame, the participant maximizes its utility by choosing CPU power usage. In the upper-level subgame, the model owner minimizes cost incurred by choosing an optimal compensation to be paid out per unit of participant's CPU power. The equilibrium solution is then solved through backward induction. The simulation results show that as budget of the model owner increases, the incentive mechanism can reduce training delay since participants are incentivized to devote more CPU power for faster training.

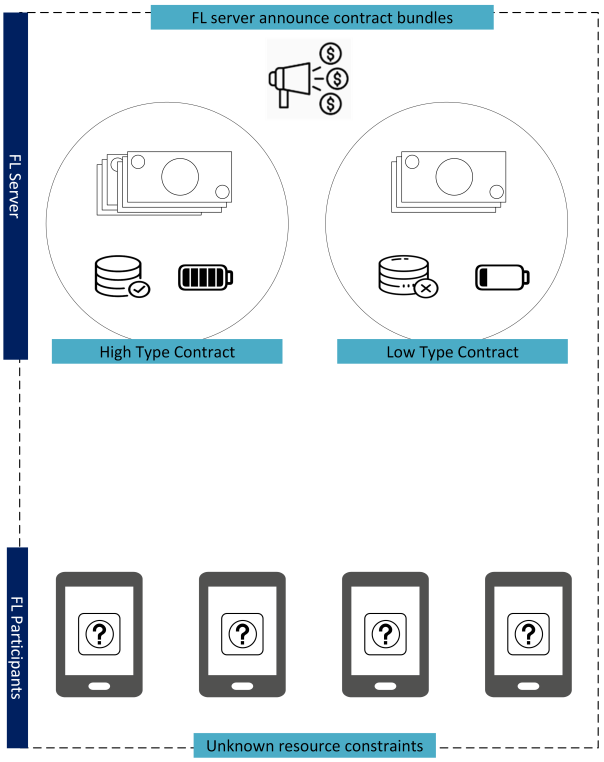

Fig. 11: Participants with unknown resource constraints maximize their utility only if they choose the bundle that best reflects their constraints.

In contrast to [130] and [132], the authors in [133] propose an incentive design using a contract theory [134] approach to attract participants with high-quality data for FL. In particular, well-designed contracts can reduce information asymmetry through self-revealing mechanisms in which participants select only the contracts specifically designed for their types. For feasibility, each contract must satisfy the Individual Rationality (IR) and Incentive Compatibility (IC) constraints. For IR, each participant is assured of a positive utility when the participant participates in the federation. For IC, every utility maximizing participant only chooses the contract designed for its type. The model owner aims to maximize its own profits subject to IR and IC constraints. As illustrated in Fig. 11, the optimal contracts derived are self-revealing such that each high-type participant with higher data quality only chooses contracts designed for its type, whereas each low-type participant with lower data quality does not have the incentive to imitate high-type participants. The simulation results show that all types of participants only achieve maximum utility when they choose the contract that matches their types. In addition, the proposed contract theory approach also has better performance in terms of profit for the model owner compared with the Stackelberg game-based incentive mechanism. This is because under the contract theoretic approach, the model owner can extract more profits from the participants whereas under the Stackelberg game approach, the participants can optimize their individual utilities.

The authors in [133] further introduce reputation as a metric to measure the reliability of FL participants and design a reputation-based participant selection scheme for reliable FL [62]. In this setting, each participant has a reputation value [135] derived from two sources, (i) direct reputation opinions from past interactions with the FL server and (ii) indirect reputation opinions from other task publishers, i.e., other FL servers. The indirect reputation opinions are stored in an open-access reputation blockchain [136] to ensure secure reputation management in a decentralized manner. Before model training, the participants choose a contract that best fits its dataset accuracy and resource conditions. Then, the FL server chooses the participants that have reputation scores which are larger than a prespecified threshold. After the FL task is completed, i.e., a desirable accuracy is achieved, the FL server updates the reputation opinions, which are subsequently stored in the reputation blockchain. The simulation results show that the proposed scheme can significantly improve the accuracy of the FL model since unreliable workers are detected and not selected for FL training.

Summary: In this section, we have discussed three main issues in resource allocation. The issues and approaches are summarized in Table IV. In sections III and IV, however, we have assumed that FL assures participants of privacy and security. However, as will be discussed in the following section, this assumption may not hold in the presence of malicious participants or FL server. As such, we proceed to discuss privacy and security issues in the following section.

V. Privacy and Security Issues

A. Privacy Issues

One of the main objectives of FL is to protect the privacy of participants, i.e., the participants only need to share parameters of the trained model instead of sharing their actual data. However, some recent research works have shown that a malicious participant can still infer sensitive information, e.g., gender, occupation, and location, from other participants based on their shared models. For example, in [137], when training a binary gender classifier on the FaceScrub [138] dataset, the authors show that they can infer if a certain participant's inputs are included in the dataset just from inspecting the shared model, with a very high accuracy of up to 90%. Thus, in this section, we discuss privacy issues related to the shared models in FL and review solutions proposed to protect the privacy of participants.

1) Information exploiting attacks in machine learning - A brief overview: One of the first research works that shows the possibility of extracting information from a trained model is [139]. In this paper, the authors show that during the training phase, the correlations implied in the training samples are gathered inside the trained model. Thus, if the trained model is released, it can lead to an unexpected information leakage to attackers. For example, an adversary can infer the ethnicity or gender of a user from its trained voice recognition system. In [140], the authors develop a model-inversion algorithm which is very effective in exploiting information from decision tree-based or face recognition trained models. The idea of this approach is to compare the target feature vector with each of the possible value and then derive a weighted probability

TABLE IV: Approaches to resource allocation in FL.

Issue	Ref.	Approach
Participant Selection	[78]	FedCS to select participants based on computation capabilities
	[114]	Hybrid-FL to select participants for IID data collection
	[115]	DRL to determine resource consumption by participants
	[119]	Fair resource allocation
	[121], [123]	Participant selection based on distance threshold to increase SNR in BAA
	[124]	Participant selection to keep signal distortion low
Adaptive Aggregation	[129], [111]	Asynchronous FL where model aggregation occurs once local updates are received by FL server
	[65]	Adaptive global aggregation frequency
Incentive Mechanism	[130]	Stackelberg game and relay network to support model update transfer
	[132]	Stackelberg game to incentivize computation resource usage in FL training
	[133], [62]	Contract theory, reputation mechanisms and blockchain

estimation which is the correct value. The experiment results reveal that by using this technique, the adversary can reconstruct an image of the victim's face from its label with a very high accuracy.

Recently, the authors in [141] show that it is even possible for an adversary to infer information of a victim through queries to the prediction model. In particular, this occurs when a malicious participant has the access to make prediction queries on a trained model. Then, the malicious participant can use the prediction queries to extract the trained model from the data owner. More importantly, the authors point out that this kind of attack can successfully extract model information from a wide range of training models such as decision trees, logistic regressions, SVMs, and even complex training models including DNNs. Some recent research works have also demonstrated the vulnerabilities of DNN-based training models against model extraction attacks [142]–[144]. Therefore, this raises a serious privacy concern for participants in sharing training models in FL.

2) Differential privacy-based protection solutions for FL participants: In order to protect the privacy of parameters trained by DNNs, the authors in [20] introduce a technique, called *differentially private stochastic gradient descent*, which can be effectively implemented on DL algorithms. The key idea of this technique is to add some "noise" to the trained parameters by using a differential privacy-preserving randomized mechanism [145], e.g., a Gaussian mechanism, before sending such parameters to the server. In particular, at the gradient averaging step of a normal FL participant, a Gaussian distribution is used to approximate the differentially private stochastic gradient descent. Then, during the training phase, the participant keeps calculating the probability that malicious participants can exploit information from its shared parameters. Once a predefined threshold is reached, the participant will stop its training process. In this way, the participant can mitigate the risk of revealing private information from its shared parameters.

Inspired by this idea, the authors in [146] develop an approach which can achieve a better privacy-protection solution for participants. In this approach, the authors propose two main steps to process data before sending trained parameters to the server. In particular, for each learning round, the aggregate server first selects a random number of participants to train the global model. Then, if a participant is selected to train the global model in a learning round, the participant will adopt the method proposed in [20], i.e., using a Gaussian distribution to add noise to the trained model before sending the trained parameters to the server. In this way, a malicious participant cannot infer information of other participants by using the parameters of shared global model as it has no information regarding who has participated in the training process in each learning round.

3) Collaborative training solutions: While DP solutions can protect private information of a honest participant from other malicious participants in FL, they only work well if the server is trustful. If the server is malicious, it can result in a more serious privacy threat to all participants in the network. Thus, the authors in [147] introduce a collaborative DL framework to render multiple participants to learn the global model without uploading their explicit training models to the server. The key idea of this technique is that instead of uploading the whole set of trained parameters to the server and updating the whole global parameters to its local model, each participant wisely selects the number of gradients to upload and the number of parameters from the global model to update as illustrated in Fig. 12. In this way, malicious participants cannot infer explicit information from the shared model. One interesting result of this paper is that even when the participants do not share all trained parameters and do not update all parameters from the shared model, the accuracy of proposed solution is still close to that of the case when the server has all dataset to train the global model. For example, for the MNIST dataset [148], the accuracy of prediction model when the participants agree to share 10% and 1% of their parameters are respectively 99.14% and 98.71%, compared with 99.17% for the centralized solution when the server has full data to train. However, the approach is yet to be tested on more complex classification tasks.

Although selective parameter sharing and DP solutions can make information exploiting attacks more challenging, the authors in [149] show that these solutions are susceptible to a new type of attack, called powerful attack, developed based on Generative Adversarial Networks (GANs) [150]. GANs is a class of ML technique which uses two neural networks,

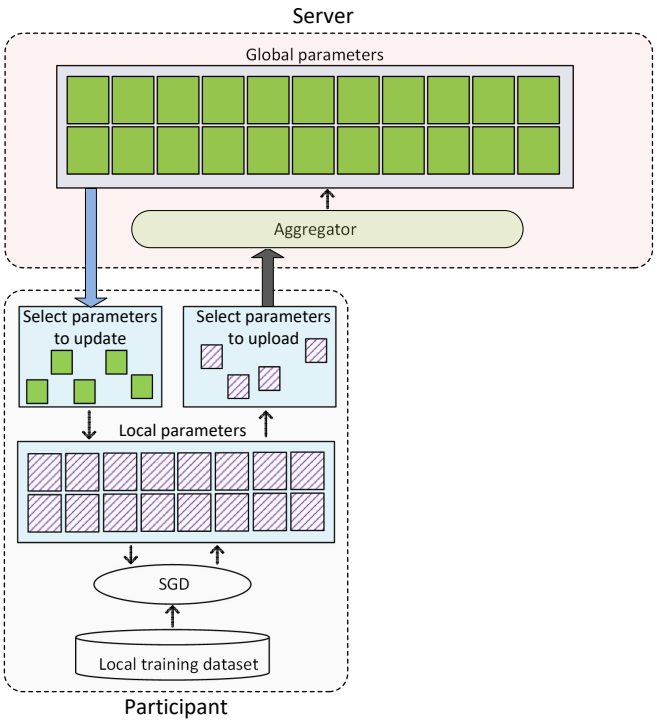

Fig. 12: Selective parameters sharing model.

namely generator network and discriminator network, that compete with each other to train data. The generator network tries to generate the fake data by adding some "noise" to the real data. Then, the generated fake data is passed to the discriminator network for classification. After the training process, the GANs can generate new data with the same statistics as the training dataset. Inspired by this idea, the authors in [149] develop a powerful attack which allows a malicious participant to infer sensitive information from a victim participant even with just a part of shared parameters from the victim as illustrated in Fig. 13. To deal with the GAN attack, the authors in [151] introduce a solution using secret sharing scheme with extreme boosting algorithm. This approach executes a lightweight secret sharing protocol before transmitting the newly trained model in plaintext to the server at each round. Thereby, other participants in the network cannot infer information from the shared model. However, the limitation of this approach is the reliance on a trusted third party to generate signature key pairs.

Different from all aforementioned works, the authors in [152] introduce a collaborative training model in which all participants cooperate to train a federated GANs model. The key idea of this method is that the federated GANs model can generate artificial data that can replace participants' real data, and thus protecting the privacy of real data for the honest participants. In particular, in order to guarantee participants' data privacy while still maintaining flexibility in training tasks, this approach produces a federated generative model. This model can output artificial data that does not belong to any real user in particular, but comes from the common cross-user data distribution. As a result, this approach can significantly reduce the possibility of malicious exploitation of information from real data. However, this approach inherits existing limitations of GANs, e.g., training instability due to the generated fake data, which can dramatically reduce the performance of collaborative learning models.

4) Encryption-based Solutions: Encryption is an effective way to protect data privacy of the participants when they want to share the trained parameters in FL. In [153], the homomorphic encryption technique is introduced to protect privacy of participants' shared parameters from a honest-but-curious server. A honest-but-curious server is defined to be a user who wants to extract information from the participants' shared parameters, but keeps all operations in FL in proper working condition. The idea of this solution is that the participants' trained parameters will be encrypted using the homomorphic encryption technique before they are sent to the server. This approach is effective in protecting sensitive information from the curious server, and also achieves the same accuracy as that of the centralized DL algorithm. A similar concept is also presented in [79] with secret sharing mechanism used to protect information of FL participants.

Although both the encryption techniques presented in [153] and [79] can prevent the curious server from extracting information, they require multi-round communications and cannot preclude collusions between the server and participants. Thus, the authors in [154] propose a hybrid solution which integrates both additively homomorphic encryption and DP in FL. In particular, before the trained parameters are sent to the server, they will be encrypted using the additively homomorphic encryption mechanism together with intentional noises to perturb the original parameters. As a result, this hybrid scheme can simultaneously prevent the curious server from exploiting information as well as solve the collusion problem between the server and malicious participants. However, in this paper, the authors do not compare the accuracy of the proposed approach with the case without homomorphic encryption and DP. Thus, the performance of proposed approach, i.e., in terms of model accuracy, is not clear.

B. Security Issues

In FL, the participants locally train the model and share trained parameters with other participants in order to improve the accuracy of prediction. However, this process is susceptible to a variety of attacks, e.g., data and model poisoning, in which a malicious participant can send incorrect parameters or corrupted models to falsify the learning process during global aggregation. Consequently, the global model will be updated incorrectly, and the whole learning system becomes corrupted. This section discusses more details on emerging attacks in FL as well as some recent countermeasures to deal with such attacks.

1) Data Poisoning Attacks: In FL, a participant trains its data and sends the trained model to the server for further processing. In this case, it is intractable for the server to check the real training data of a participant. Thus, a malicious participant can poison the global model by creating *dirty-label* data to train the global model with the aim of generating falsified parameters. For example, a malicious participant can

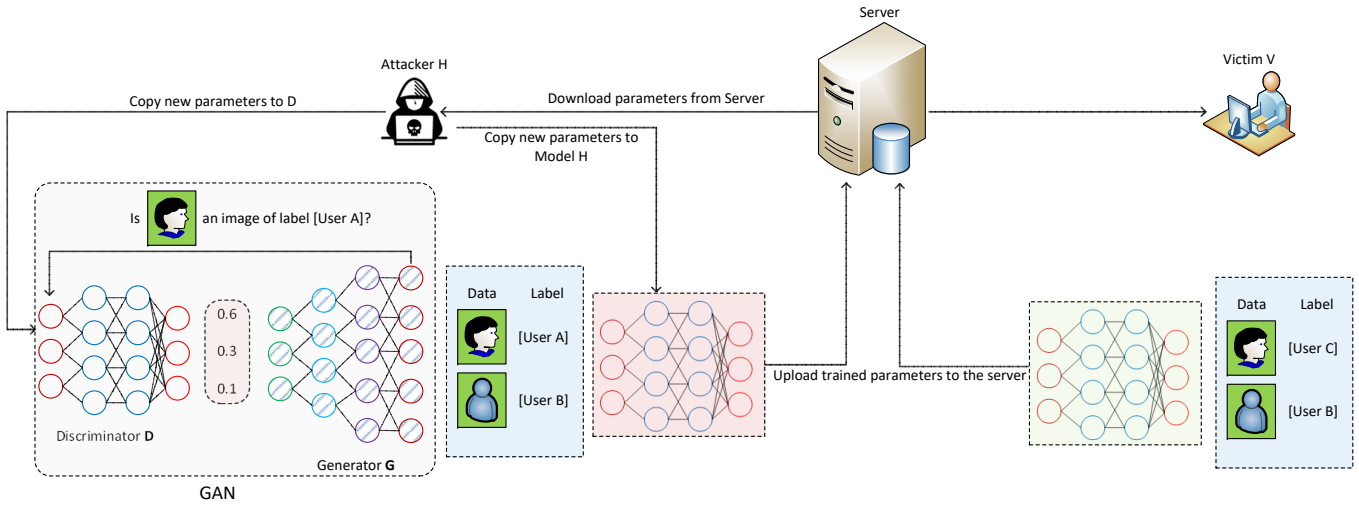

Fig. 13: GAN Attack on collaborative deep learning.

generate a number of samples, e.g., photos, under a designed label, e.g., a clothing branch, and use them to train the global model to achieve its business goal, e.g., the prediction model shows results of the targeted clothing branch. Dirty-label data poisoning attacks are demonstrated to achieve high misclassifications in DL processes, up to 90%, when a malicious participant injects relatively few dirty-label samples (around 50) to the training dataset [155]. This calls for urgent solutions to deal with data poisoning attacks in FL.

In [156], the authors investigate impacts of a sybil-based data poisoning attack to a FL system. In particular, for the sybil attack, a malicious participant tries to improve the effectiveness of data poisoning in training the global model by creating multiple malicious participants. In Table V, the authors show that with only two malicious participants, the attack success rate can achieve up to 96.2%, and now the FL model is unable to correctly classify the image of "1" (instead it always incorrectly predicts them to be the image of "7"). To mitigate sybil attacks, the authors then propose a defense strategy, namely FoolsGold. The key idea of this approach is that honest participants can be distinguished from sybil participants based on their updated gradients. Specifically, in the non-IID FL setting, each participant's training data has its own particularities, and sybil participants will contribute gradients that appear more similar to each other than those of other honest participants. With FoolsGold, the system can defend the sybil data poisoning attack with minimal changes to the conventional FL process and without requiring any auxiliary information outside of the learning process. Through simulations results on 3 diverse datasets (MNIST [148], KDDCup [157], Amazon Reviews [157]), the authors show that FoolsGold can mitigate the attack under a variety of conditions, including different distributions of participant data, varying poisoning targets, and various attack strategies.

2) Model Poisoning Attacks: Unlike data poisoning attacks which aim to generate fake data to cause adverse impacts to the global model, a model poisoning attack attempts to directly poison the global model that it sends to the server for aggregation. As shown in [158] and [159], model poisoning

TABLE V: The accuracy and attack success rates for no-attack scenario and attacks with 1 and 2 sybils in a FL system with MNIST dataset [148].

	Baseline	**Attack 1**	**Attack 2**
Number of honest participants	10	10	10
Number of sybil participants	0	1	2
The accuracy (digits: 0, 2-9)	90.2%	89.4%	88.8%
The accuracy (digit: 1)	96.5%	60.7%	0.0%
Attack success rate	0.0%	35.9%	96.2%

attacks are much more effective than those of data poisoning attacks, especially for large-scale FL with many participants. The reason is that for data poisoning attacks, a malicious participant's updates are scaled based on its dataset and the number of participants in the federation. However, for model poisoning attacks, a malicious participant can modify the updated model, which is sent to the server for aggregation, directly. As a result, even with one single attacker, the whole global model can be poisoned. The simulation results in [158] also confirm that even a highly constrained adversary with limited training data can achieve high success rate in performing model poisoning attacks. Thus, solutions to protect the global model from model poisoning attacks have to be developed.

In [158], some solutions are suggested to prevent model poisoning attacks. Firstly, based on an updated model shared from a participant, the server can check whether the shared model can help to improve the global model's performance or not. If not, the participant will be marked to be a potential attacker, and after few rounds of observing the updated model from this participant, the server can determine whether this is a malicious participant or not. The second solution is based on the comparison among the updated models shared by the participants. In particular, if an updated model from a participant is too different from the others, the participant can potentially be a malicious one. Then, the server will continue observing updates from this participant before it can determine whether this is a malicious user or not. However, model poisoning attacks are extremely difficult to prevent because when training with millions of participants, it is intractable

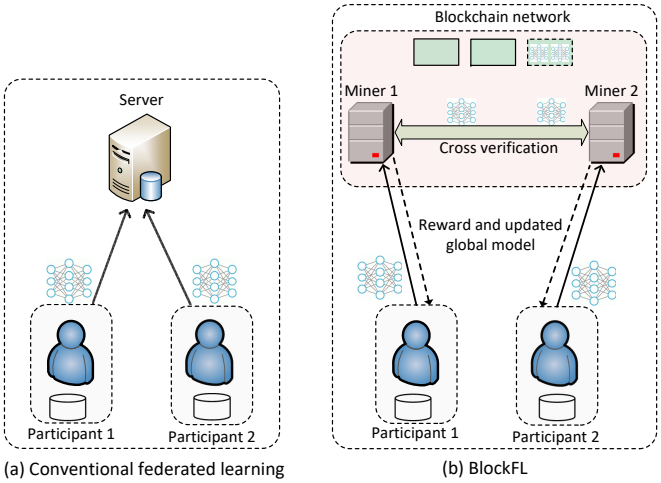

Fig. 14: An illustration of (a) conventional FL and (b) the proposed BlockFL architectures.

to evaluate the improvement from every single participant. As such, more effective solutions need to be further investigated.

In [159], the authors introduce a more effective model poisoning attack which is demonstrated to achieve 100% accuracy on the attacker's task within just a single learning round. In particular, a malicious participant can share its poisoned model which not only is trained for its intentional purpose, but which also contains a backdoor function. In this paper, the authors consider to use a semantic backdoor function to inject into the global model. The reason is that this function can make the global model misclassify even without a need to modify the input data of the malicious participant. For example, an image classification backdoor function can inject an attacker-chosen label to all images with some certain features, e.g., all dogs with black stripes can be misclassifed to be cats. In the simulations, the authors show that this attack can greatly outperform other conventional FL data poisoning attacks. For example, in a word-prediction task with 80,000 total participants, compromising just eight of them is enough to achieve 50% backdoor accuracy, as compared to 400 malicious participants needed to perform the data-poisoning attack.

3) Free-Riding Attacks: Free-riding is another attack in FL that occurs when a participant wants to benefit from the global model without contributing to the learning process. The malicious participant, i.e., free-rider, can pretend that it has very small number of samples to train or it can select a small set of its real dataset to train, e.g., to save its resources. As a result, the honest participants need to contribute more resources in the FL training process. To address this problem, the authors in [160] introduce a blockchain-based FL architecture, called BlockFL, in which the participants' local learning model updates are exchanged and verified by leveraging the blockchain technology. In particular, each participant trains and sends the trained global model to its associated miner in the blockchain network and then receives a reward that is proportional to the number of trained data samples as illustrated in Fig. 14. In this way, this framework can not only prevent the participants from free-riding, but also incentivize all participants to contribute to the learning process. A similar blockchain-based model is also introduced in [161] to provide data confidentiality, computation auditability, and incentives for the participants of FL. However, the utilization of the blockchain technology implies the incurrence of a significant cost for implementing and maintaining miners to operate the blockchain network. Furthermore, consensus protocols used in blockchain networks, e.g., proof-of-work (PoW), can cause a long delay in information exchange, and thus they may not be appropriate to implement on FL models.

Summary: In this section, we have discussed two key issues, i.e., privacy and security, when trained models are exchanged in FL. In general, it is believed that FL is an effective privacy-preserving learning solution for participants to perform collaborative model training. However, in this section, we show that a malicious participant can exploit the process and gain access to sensitive information of other participants. Furthermore, we have also shown that by using the shared model in FL, an attacker can perform attacks which can not only breakdown the whole learning system, but also falsify the trained model to achieve its malicious goal. In addition, solutions to deal with these issues have also been reviewed, which are especially important in order to guide FL system administrators in designing and implementing appropriate countermeasures. Key information of attacks and their corresponding countermeasures in FL are summarized in Table VI.

VI. Applications of Federated Learning for Mobile Edge Computing

In the aforementioned studies, we have discussed the issues pertaining to the implementation of FL as an enabling technology that allows collaborative learning at mobile edge networks. In this section, we focus instead on the applications of FL for mobile edge network optimization.

As highlighted by the authors in [34], the increasing complexity and heterogeneity of wireless networks enhance the appeal of adopting a data-driven ML based approach [27] for optimizing system designs and resource allocation decision making for mobile edge networks. However, as discussed in previous sections, the private data of users may be sensitive in nature. As such, existing learning based approach can be combined with FL for privacy-preserving applications.

In this section, we consider four applications of FL in edge computing:

- *Cyberattack Detection:* The ubiquity of IoT devices and increasing sophistication of cyberattacks [162] imply that there is a need to improve existing cyberattack detection tools. Recently, DL has been widely successful in cyber-attack detection. Coupled with FL, cyberattack detection models can be learned collaboratively while maintaing user privacy.
- *Edge Caching and Computation Offloading:* Given the computation and storage capacity constraints of edge servers, some computationally intensive tasks of end devices have to be offloaded to the remote cloud server

TABLE VI: A summary of attacks and countermeasures in FL.

Attack Types	Attack Method	Countermeasures
Information exploiting attacks (privacy issues)	Attackers try to illegally exploit information from the shared model.	• *Differentially private stochastic gradient descent*: Add "noise" to the trained parameters by using a differential privacy-preserving randomized mechanism [20]. • *Differentially private and selective participants*: Add "noise" to the trained parameters and select randomly participants to train global model in each round [146]. • *Selective parameter sharing*: Each participant wisely selects the number of gradients to upload and the number of parameters from the global model to update [147]. • *Secrete sharing scheme with extreme boosting algorithm*: This approach executes a lightweight secret sharing protocol before transmitting the newly trained model in plaintext to the server at each round [151]. • *GAN model training*: All participants are cooperative to train a federated GANs model [152].
Data poisoning attacks	Attackers poison the global model by creating *dirty-label* data and use such data to train the global model.	• *FoolsGoal*: Distinguish honest participants based on their updated gradients. It is based on the fact that in the non-IID FL setting, each participant's training data has its own particularities, and malicious participants will contribute gradients that appear more similar to each other than those of the honest participants [156].
Model poisoning attacks	Attackers attempt to directly poison the global model that they send to the server for aggregation.	• Based on an updated model shared from a participant, the server can check whether the shared model can help to improve the global model's performance or not. If not, the participant will be marked to be a potential attacker [158]. • Compare among the updated global models shared by the participants, and if an updated global model from a participant is too different from others, it could be a potential malicious participant [158].
Free-riding attacks	Attackers benefit from the global model without contributing to the learning process, e.g., by pretending that they have very small number of samples to train.	• *BlockFL*: Participants' local learning model updates are exchanged and verified by leveraging blockchain technology. In particular, each participant trains and sends the trained global model to its associated miner in the blockchain network and then receives a reward that is proportional to the number of trained data samples [160].

for computation. In addition, commonly requested files or services should be placed on edge servers for faster retrieval, i.e., users do not have to communicate with the remote cloud when they want to access these files or services. As such, an optimal caching and computation offloading scheme can be collaboratively learned and optimized with FL.

- *Base Station Association:* In a dense network, it is important to optimize base station association so as to limit interference faced by users. However, traditional learning based approaches that utilize user data often assume that such data is centrally available. Given user privacy constraints, an FL based approach can be adopted instead.
- *Vehicular Networks:* The Internet of Vehicles (IoV) [163] features smart vehicles with data collection, computation and communication capabilities for relevant functions, e.g., navigation and traffic management. However, this wealth of knowledge is again, private and sensitive in nature since it can reveal the driver's location and personal information. In this section, we discuss the use of an FL based approach in traffic queue length prediction and energy demand in electric vehicle charging stations done at the edge of IoV networks.

A. Cyberattack Detection

Cyberattack detection is one of the most important steps to promptly prevent and mitigate serious consequences of attacks in mobile edge networks. Among different approaches to detect cyberattacks, DL is considered to be the most effective tool to detect a wide range of attacks with high accuracy. In [173], the authors show that DL can outperform all conventional ML techniques with very high accuracy in detecting intrusions on three datasets, i.e., KDDcup 1999, NSL-KDD [174], and UNSW-NB15 [175]. However, the detection accuracy of solutions based on DL depends very much on the available datasets. Specifically, DL algorithm only can outperform other ML techniques when given sufficient data to train. However, this data may be sensitive in nature. Therefore, some FL-based attack detection models for mobile edge networks have been introduced recently to address this problem.

In [164], the authors propose a cyberattack detection model for an edge network empowered by FL. In this model, each edge node operates as a participant who owns a set of data for intrusion detection. To improve the accuracy in detecting attacks, after training the global model, each participant will send its trained model to the FL server. The server will aggregate all parameters from the participants and send the updated global model back to all the participants as illustrated in Fig. 15. In this way, each edge node can learn from other edge nodes without a need of sharing its real data. As a result, this method can not only improve accuracy in detecting attacks, but also enhance the privacy of intrusion data at the edge nodes and reduce traffic load for the whole network. A similar idea is also presented in [165] in which IoT gateways operate as FL participants and an IoT security service provider works as a server node to aggregate trained models shared by the participants. The authors in [165] show empirically that by using FL, the system can successfully detect 95.6% of attacks in approximately 257 ms without raising any false alarm when evaluated in a real-world smart home deployment setting.

In both [164] and [165], it is assumed that the participants, i.e., edge nodes and IoT gateways, are honest, and they are willing to contribute in training their updated model parameters. However, if some of the participants are malicious, they can make the whole intrusion detection corrupted. Thus, the authors in [166] propose to use blockchain technology in managing data shared by the participants. By using the

TABLE VII: FL based approaches for mobile edge network optimization.

Applications	Ref.	Description
Cyberattack Detection	[164]	Cyberattack detection with edge nodes as participants
	[165]	Cyberattack detection with IoT gateways as participants
	[166]	Blockchain to store model updates
Edge caching and computation offloading	[31]	DRL for caching and offloading in UEs
	[167]	DRL for computation offloading in IoT devices
	[168]	Stacked autoencoder learning for proactive caching
	[169]	Greedy algorithm to optimize service placement schemes
Base station assoication	[170]	Deep echo state networks for VR application
	[30]	Mean field game with imitation for cell association
Vehicular networks	[171]	Extreme value theory for large queue length prediction
	[172]	Energy demand learning in electric vehicular networks

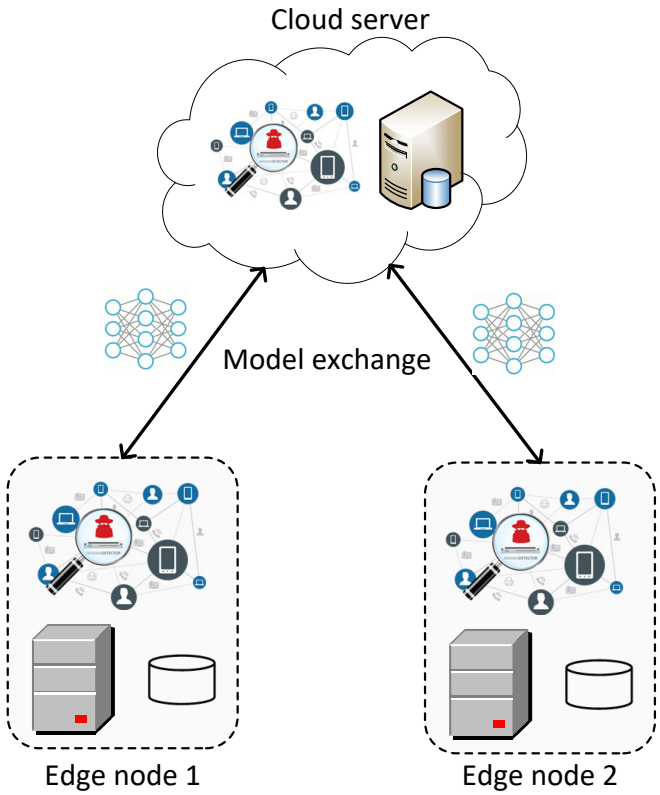

Fig. 15: FL-based attack detection architecture for IoT edge networks.

blockchain, all incremental updates to the anomaly detection ML model are stored in the ledger, and thus a malicious participant can be easily identified. Furthermore, based on shared models from honest participants stored in the ledger, the intrusion detection system can easily recover the proper global model if the current global model is poisoned.

B. Edge Caching and Computation Offloading

To account for the dynamic and time-varying conditions in a MEC system, the authors in [31] propose the use of DRL with FL to optimize caching and computation offloading decisions in an MEC system. The MEC system consists of a set of user equipments (UEs) covered by base stations. For caching, the DRL agent makes the decision to cache or not to cache the downloaded file, and which local file to replace should caching occur. For computation offloading, the UEs can choose to either offload computation tasks to the edge node via wireless channels, or perform the tasks locally. This caching and offloading decision process is illustrated in Fig. 16. The states of the MEC system include wireless network conditions, UE energy consumption, and task queuing states, whereas the reward function is defined as quality of experience (QoE) of the UEs. Given the large state and action space in the MEC environment, a DDQN approach is adopted. To protect the privacy of users, an FL approach is proposed in which training can occur with data remaining on the UEs. In addition, existing FL algorithms, e.g., *FedAvg* [23], can also ensure that training is robust to the unbalanced and non-IID data of the UEs. The simulation results show that the DDQN with FL approach achieves similar average utilities among UEs as compared to the centralized DDQN approach, while consuming less communication resources and preserving user privacy. However, the simulations are only performed with 10 UEs. If the implementation is expanded to target a larger number of heterogeneous UEs, there can be significant delays in the training process especially since the training of a DRL model is computationally intensive. As an extension, transfer learning [176] can be used to increase the efficiency of training, i.e., training is not initialized from scratch.

Similar to [31], the authors in [167] propose the use of DRL in optimizing computation offloading decisions in IoT systems. The system model consists of IoT devices and edge nodes. The IoT devices can harvest energy units [177] from the edge nodes to be stored in the energy queue. In addition, an IoT device also maintains a local task queue with unprocessed and unsuccessfully processed tasks. These tasks can be processed locally or offloaded to the edge nodes for processing, in a First In First Out (FIFO) order [14]. In the DRL problem formulation, the network states are defined to be a function of energy queue length, task execution delay, task handover delay from edge node association, and channel gain between the IoT device and edge nodes. A task can fail to be executed, e.g., when there is insufficient energy units or communication bandwidth for computation offloading. The utility considered is a function of task execution delay, task queuing delay, number of failed tasks and penalty of execution failure. The DRL agent makes the decision to either offload computation to the edge nodes or perform computation locally. To ensure privacy of users, the agent is trained without users having to upload their own data to a centralized server. In each training round, a random set of IoT devices are selected to download the model

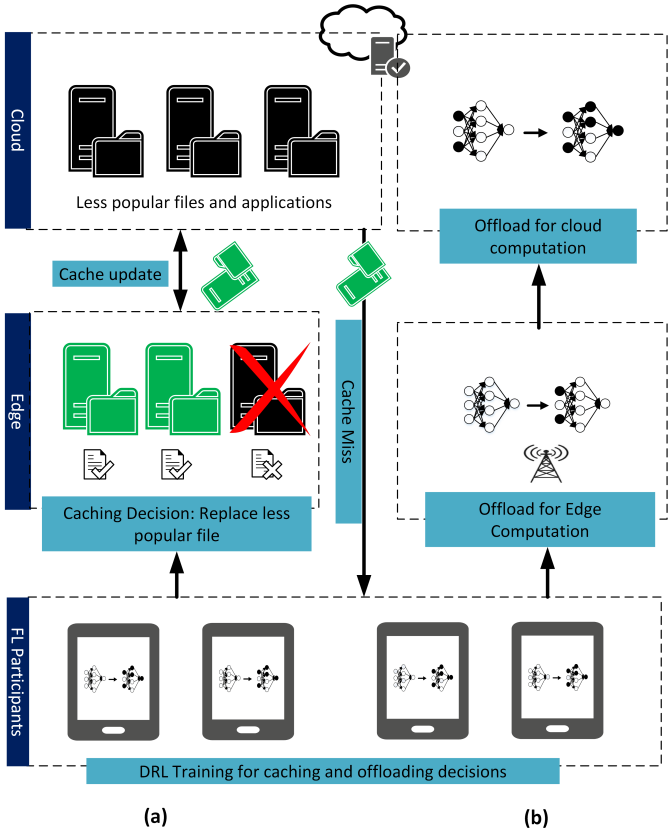

Fig. 16: FL-based (a) caching and (b) computation offloading.

parameters of the DRL agent from the edge networks. The model parameters are then updated using their own data, e.g., energy resource level, channel gain, and local sensing data. Then, the updated parameters of the DRL agent are sent to the edge nodes for model aggregation. The simulation results show that the FL based approach can achieve same levels of total utility as the centralized DRL approach. This is robust to varying task generation probabilities. In addition, when task generation probabilities are higher, i.e., there are more tasks for computation in the IoT device, the FL based scheme can achieve a lower number of dropped tasks and shorter queuing delay than the centralized DRL scheme. However, the simulation only involves 15 IoT devices serviced by relatively many edge nodes. To better reflect practical scenarios where fewer edge nodes have to cover several IoT devices, further studies can be conducted on optimizing the edge-IoT collaboration. For example, the limited communication bandwidth can cause significant task handover delay during computation offloading. In addition, with more IoT devices, the DRL training will take a longer time to converge especially since the devices have heterogeneous computation capabilities.

Instead of using a DRL approach, the authors in [168] propose the use of an FL based stacked autoencoder learning model, i.e., FL based proactive content caching scheme (FPCC), to predict content popularity for optimized caching while protecting user privacy. In the system model, each user is equipped with a mobile device that connects to the base station that covers its geographical location. Using a stacked autoencoder learning model, the latent representation of a user's information, e.g., location, and file rating, i.e., content request history, is learned. Then, a similarity matrix between the user and its historically requested files is obtained in which each element of the matrix represents the distance between the user and the file. Based on this similarity matrix, the K nearest neighbours of each user are determined, and the similarity between the user's historical watch list and the neighbours' are computed. An aggregation approach is then used to predict the most popular files for caching, i.e., files with highest similarity scores across all users. Being the most popular files across users that are most frequently retrieved, the cached files need not be re-downloaded from its source server everytime it is demanded. To protect the privacy of users, FL is adpoted to learn the parameters of the stacked autoencoder without the user having to reveal its personal information or its content request history to the FL server. In each training round, the user first downloads a global model from the FL server. Then, the model is trained and updated using their local data. The updated models are subsequently uploaded to the FL server and aggregated using the *FedAvg* algorithm. The simulation results show that the proposed FPCC scheme could achieve the highest cache efficiency, i.e, the ratio of cached files matching user requests, as compared to other caching methods such as the Thompson sampling methods [178]. In addition, privacy of the user is preserved.

The authors in [169] introduce a privacy-aware service placement scheme to deploy user-preferred services on edge servers with consideration for resource constraints in the edge cloud. The system model consists of a mobile edge cloud serving various mobile devices. The user's preference model is first built based on information such as number of times of requests for a service, and other user context information, e.g., ages and locations. However, since this can involve sensitive personal information, an FL based approach is proposed to train the preference model while keeping users' data on their personal devices. Then, an optimization problem is formulated in which the objective is to maximize quantity of services demanded from the edge based on user preferences, subject to constraints of storage capacity, computation capability, uplink and downloading bandwidth. The optimization problem is then solved using a greedy algorithm, i.e., the service which most improves the objective function is added till resource constraints are met. The simulation results show that the proposed scheme can outperform the popular service placement scheme, i.e., where only the most popular services are placed on the edge cloud, in terms of number of requests processed on edge clouds since it also considers the aforementioned resource constraints in maximizing quantity of services.

C. Base Station Association

The authors in [170] propose an FL based deep echo state networks (ESNs) approach to minimize breaks in presence (BIPs) [179] for users of virtual reality (VR) applications. A BIP event can be a result of delay in information transmission which can be caused when the user's body movements obstruct the wireless link. BIPs cause the user to be aware that they are

in a virtual environment, thus reducing their quality of experience. As such, a user association policy has to be designed such that BIPs are minimized. The system model consists of base stations that cover a set of VR users. The base stations receive uploaded tracking information from each associated user, e.g., physical location and orientation, while the users download VR videos for their use in the VR application. For data transmission, the VR users have to associate with one of the base stations. As such, a minimization problem is formulated where BIPs are minimized with respect to expected locations and orientations of the VR user. To derive a prediction of user locations and orientations, the base station has to rely on the historical information of users. However, the historical information stored at each base station only collects partial data from each user, i.e., a user connects to multiple base stations and its data is distributed across them. As such, an FL based approach is implemented whereby each base station first trains a local model using its partial data. Then, the local models are aggregated to form a global model capable of generalization, i.e., comprehensively predicting a user's mobility and orientations. The simulation results show that the federated ESN algorithm can achieve lower BIPs experienced by users as compared to the centralized ESN algorithm proposed in [180], since a centralized approach only makes partial prediction with the incomplete data from sole base stations, whereas the federated ESN approach can make predictions based on a model learned collaboratively from more complete data.

The authors in [30] also consider solving the problem of cell association in dense wireless networks with a collaborative learning approach. In the system model, the base stations cover a set of users in an LTE cellular system. In a cellular system, users are likely to face similar channel conditions as their neighbors and thus can benefit from learning from their neighbours that are already associated with base stations. As such, the cell association problem is formulated as a mean-field game (MFG) with imitation [181] in which each user maximizes its own throughput while minimizing the cost of imitation. The MFG is further reduced into a single-user Markov decision process that is then solved by a neural Q-learning algorithm. In most other proposed solution for cell association, it is assumed that all information is known to the base stations and users. However, given privacy concerns, the assumption of information sharing may not be practical. As such, a collaborative learning approach can be considered where only the outcome of the learning algorithm is exchanged during the learning process whereas usage data is kept locally in each user's own device. The simulation results show that imitating users can attain higher utility within a shorter training duration as compared to non-imitating users.

D. Vehicular Networks

Ultra reliable low latency communication (URLLC) in vehicular networks is an essential prerequisite towards developing an intelligent transport system. However, existing radio resource management techniques do not account for rare events such as large queue lengths at the tail-end distribution. To model the occurence of such low probability events, the authors in [171] propose the use of extreme value theory (EVT) [182]. The approach requires sufficient samples of queue state information (QSI) and data exchange among vehicles. As such, an FL approach is proposed in which vehicular users (VUEs) train the learning model with data kept locally and upload only their updated model parameters to the roadside units (RSU). The RSU then averages out the model parameters and return an updated global model to the VUEs. In a synchronous approach, all VUEs upload their models at the end of a prespecified interval. However, the simultaneous uploading by multiple vehicles can lead to delays in communication. In contrast for an asynchronous approach, each VUE only evaluates and uploads their model parameters after a predefined number of QSI samples are collected. The global model is also updated whenever a local update is received, thus reducing communication delays. To further reduce overhead, Lyapunov optimization [183] for power allocation is also utilized. The simulation results show that under this framework, there is a reduction of the number of vehicles experiencing large queue lengths whereas FL can ensure minimal data exchange relative to a centralized approach.

The authors in [172] propose a federated energy demand learning (FEDL) approach to manage energy resources in charging stations (CSs) for electric vehicles (EVs). When a large number of EVs congregate at a CS, this can lead to energy transfer congestion. To resolve this, energy is supplied from the power grids and reserved in advance to meet the real-time demands from the EVs [184], rather than having the CSs request for energy from the power grid only upon receiving charging requests. As such, there is a need to forecast energy demand for EV networks using historical charging data. However, this data is usually stored separately at each of the CS that the EVs utilize and is private in nature. As such, an FEDL approach is adopted in which each CS trains the demand prediction model on its own dataset before sending only the gradient information to the charging station provider (CSP). Then, the gradient information from the CS is aggregated for global model training. To further improve model accuracy, the CSs are clustered using the constrained K-means algorithm [185] based on their physical locations. The clustering-based FEDL reduces the cost of biased prediction [186]. The simulation results show that the root mean squared error of a clustered FEDL model is lower than conventional ML algorithms, e.g., multi-layer perceptron regressor [187]. However, the privacy of user data is still not protected by this approach, since user data is stored in each of the CS. As an extension, the user data can possibly be stored in each EVs separately, and model training can be conducted in the EVs rather than the CSs. This can allow more user features to be considered to enhance the accuracy of EDL, e.g., user consumption habits.

Summary: In this section, we discuss that FL can also be used for mobile edge network optimization. In particular, DL and DRL approaches are suitable for modelling the dynamic environment of increasingly complex edge networks but require sufficient data for training. With FL, model training can be carried out while preserving the privacy of users. A

summary of the approaches are presented in Table VII.

VII. Challenges, Open Issues, and Future Research Directions

Apart from the aforementioned issues, there are still challenges, open issues, and new research directions in deploying FL at scale to be discussed as follows.

A. Challenges

1) Dropped participants: The approaches discussed in Section IV, e.g., [78], [114], and [115], propose new algorithms for participant selection and resource allocation to address the training bottleneck and resource heterogeneity. In these approaches, the wireless connections of participants are assumed to be always available. However, in practice, participating mobile devices may go offline and can drop out from the FL system due to connectivity or energy constraints. A large number of dropped devices from the training participation can significantly degrade the performance [23], e.g., accuracy and convergence speed, of the FL system. New FL algorithms need to be robust to device drop out in the networks and anticipate the scenarios in which only a small number of participants are left connected to participate in a training round. One potential solution is that the FL model owner provides free dedicated/special connection, e.g., cellular connections, as an incentive to the participants to avoid drop out.

2) Privacy concerns: FL is able to protect the privacy of each participants since the model training may be conducted locally, with just the model parameters exchanged with the FL server. However, as specified in [139], [140], and [141], communicating the model updates during the training process can still reveal sensitive information to an adversary or a third-party. The current approaches propose security solutions such as DP, e.g., [20], [146], and [188], and collaborative training, e.g., [147] and [149]. However, the adoption of these approaches sacrifices the performance, i.e., model accuracy. They also require significant computation on participating mobile devices. Thus, the tradeoff between privacy guarantee and system performance has to be well balanced when implementing the FL system.

3) Unlabeled data: It is important to note that the approaches reviewed in the survey are proposed for supervised learning tasks. This means that the approaches assume that labels exist for all of the data in the federated network. However, in practice, the data generated in the network may be unlabeled or mislabeled [189]. This poses a big challenge to the server to find participants with appropriate data for model training. Tackling this challenge may require the challenges of scalability, heterogeneity, and privacy in the FL systems to be addressed. One possible solution is to enable mobile devices to construct their labeled data by learning the "labeled data" from each other.

B. Open Issues

1) Interference among mobile devices: The existing resource allocation approaches, e.g., [78] and [115], address the participant selection based on the resource states of their mobile devices. In fact, these mobile devices may be geographically close to each other, i.e., in the same cell. This introduces an interference issue when they update local models to the server. As such, channel allocation policies may need to be combined with the resource allocation approaches to address the interference issue. While studies in [121], [123], and [124] consider multi-access schemes and over-the-air computation, it remains to be seen if such approaches are scalable, i.e., able to support a large federation of many participants. To this end, data driven learning based solutions, e.g., federated DRL, can be considered to model the dynamic environment of mobile edge networks and make optimized decisions.

2) Communication security: Due to the exposed nature of the wireless medium, FL is vulnerable to serious security issues such as Distributed Denial-of-Service (DoS) [190] and jamming attack [191]. In particular for jamming attacks, an attacker can transmit radio frequency jamming signals with high power to disrupt or cause interference to the communications between the mobile devices and the server. Such an attack can cause errors to the model uploads/downloads and consequently degrade the performance, i.e., accuracy, of the FL systems. Anti-jamming schemes [192] such as frequency hopping, e.g., sending one more copy of the model update over different frequencies, can be adopted to address the issue.

3) Asynchronous FL: In synchronous FL, each training round only progresses as quickly as the slowest device, i.e., the FL system is susceptible to the stragglers effect. As such, asynchronous FL has been proposed as a solution in [111] and [129]. In addition, asynchronous FL also allows participants to join the FL training halfway even while a training round is in progress. This is more reflective of practical FL settings and can be an important contributing factor towards ensuring the scalability of FL. However, synchronous FL remains to be the most common approach used due to convergence guarantees [77]. Given the many advantages of asynchronous FL, new asynchronous algorithms should be explored. In particular, for future proposed algorithms, the convergence guarantee in a non-IID setting for non-convex loss functions need to be considered.

4) Incentive mechanism designs: The incentive mechanism designs proposed in [130], [132] and [133] assume that a federation consists only of multiple individual participants, i.e., solo FL with one FL server. There can be exceptions to this setting as follows: (i) the participants may be competitors who are reluctant to share their model parameters since competitors also benefit from a trained global model (ii) the FL servers may be competing with other FL servers, i.e., model owners. In this case, the formulation of the incentive mechanism design will be vastly different from that proposed. In addition, other mechanisms can also be adopted, e.g., auctions [193], [194].

C. Future Directions

1) New studies on learning convergence: One of the essential considerations of FL is the convergence of the algorithm. FL finds weights to minimize the global model aggregation. This is actually a distributed optimization problem, and the

convergence is not always guaranteed. Theoretical analysis and evaluations on the convergence bounds of the gradient descent based FL for convex and non-convex loss functions are important research directions. While existing studies have covered this topic, many of the guarantees are limited to restrictions, e.g., convexity of the loss function.

2) New tools for quantifying statistical heterogeneity: Mobile devices typically generate and collect data in a non-IID manner across the network. Moreover, the number of data samples among the mobile devices may vary significantly. To improve the convergence of FL algorithm, the statistical heterogeneity of the data needs to be quantified. Recent works, e.g., [195], have developed tools for quantifying statistical heterogeneity through metrics such as local dissimilarity. However, these metrics cannot be easily calculated over the federated network before training begins. The importance of these metrics motivates future directions such as the development of efficient algorithms to quickly determine the level of heterogeneity in federated networks.

3) Combined algorithms for communication reduction: Currently, there are three common techniques of communication reduction in FL as discussed in Section III. It is important to study how these techniques can be combined with each other to improve the performance further. For example, the model compression technique can be combined with the importance-based updating technique. The combination is able to significantly reduce the size of model updates sent from the mobile devices to the server. However, the trade-off between accuracy and communication overhead for the combination technique needs to be further evaluated. In particular, for simulation results we discuss in Section III, the accuracy-communication cost reduction tradeoff is difficult to manage since it varies for different settings, e.g., datasets and number of participants.

4) Cooperative mobile crowd ML: In the existing approaches, mobile devices needs to communicate with the server directly and this may increase the energy consumption. In fact, mobile devices nearby can be grouped in a cluster, and the model downloading/uploading between the server and the mobile devices can be facilitated by a "cluster head" that serves as a relay node. [196]. The model exchange between the mobile devices and the cluster head can be done in Device-to-Device (D2D) connections. Such a model can improve the energy efficiency significantly. Efficient coordination schemes for the cluster head can thus be designed to further improve the energy efficiency of a FL system.

5) Applications of FL: Given the advantages of guaranteeing data privacy, FL has an increasingly important role to play in many applications, e.g., healthcare, finance and transport systems. For most current studies on FL applications, the focus seems to be on federated training of the learning model, with the implementation challenges neglected. For future studies on the applications of FL, there is a need to consider the aforementioned issues in the survey, i.e., communication costs, resource allocation, and privacy and security to ensure that a FL system is feasible, well-designed and scalable.

VIII. Conclusion

This paper has presented a tutorial of FL and a comprehensive survey on the issues regarding FL implementation. Firstly, we begin with an introduction to the motivation for MEC, and how FL can serve as an enabling technology for collaborative model training at mobile edge networks. Then, we describe the fundamentals of DNN model training, FL, and system design towards FL at scale. Afterwards, we provide detailed reviews, analyses, and comparisons of approaches for emerging implementation challenges in FL. The issues include communication cost, resource allocation, data privacy and data security. Furthermore, we also discuss the implementation of FL for privacy-preserving mobile edge network optimization. Finally, we discuss challenges, open issues, and future research directions.

References

[1] K. L. Lueth, "State of the iot 2018: Number of iot devices now at 7b Ð market accelerating," *IOT Analytics*, 2018.

[2] R. Pryss, M. Reichert, J. Herrmann, B. Langguth, and W. Schlee, "Mobile crowd sensing in clinical and psychological trials–a case study," in *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. IEEE, 2015, pp. 23–24.

[3] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[5] D. Oletic and V. Bilas, "Design of sensor node for air quality crowd-sensing," in *2015 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2015, pp. 1–5.

[6] Y. Jing, B. Guo, Z. Wang, V. O. Li, J. C. Lam, and Z. Yu, "Crowd-tracker: Optimized urban moving object tracking using mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3452–3463, 2017.

[7] H.-J. Hong, C.-L. Fan, Y.-C. Lin, and C.-H. Hsu, "Optimizing cloud-based video crowdsensing," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 299–313, 2016.

[8] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587–1595, 2014.

[9] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.

[10] B. Custers, A. Sears, F. Dechesne, I. Georgieva, T. Tani, and S. van der Hof, *EU Personal Data Protection in Policy and Practice*. Springer, 2019.

[11] B. M. Gaff, H. E. Sussman, and J. Geetter, "Privacy and big data," *Computer*, vol. 47, no. 6, pp. 7–9, 2014.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[13] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[14] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[15] Y. Han, X. Wang, V. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *arXiv preprint arXiv:1907.08349*, 2019.

[16] Cisco, "Cisco global cloud index: Forecast and methodology, 2016Ð2021 white paper."

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[18] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[20] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[21] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," 2016.

[22] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, "Edge computing and social internet of things for large-scale smart environments development," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2557–2571, 2017.

[23] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[24] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[25] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[26] D. Verma, S. Julier, and G. Cirincione, "Federated ai for building ai solutions across multiple agencies," *arXiv preprint arXiv:1809.10036*, 2018.

[27] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[28] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[29] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[30] K. Hamidouche, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Collaborative artificial intelligence (ai) for user-cell association in ultra-dense cellular systems," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[31] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *arXiv preprint arXiv:1809.07857*, 2018.

[32] S. Samarakoon, M. Bennis, W. Saady, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *arXiv preprint arXiv:1807.08127*, 2018.

[33] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[34] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," *arXiv preprint arXiv:1908.06847*, 2019.

[35] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[36] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *arXiv preprint arXiv:1905.10083*, 2019.

[37] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for internet of things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[38] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[39] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[40] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[41] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[42] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[43] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514–525, 2014.

[44] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 5200–5204.

[45] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[46] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[47] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.

[48] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[49] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, p. 8, 2012.

[50] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[51] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[52] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[53] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[55] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[56] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[57] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[58] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.

[59] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119–135, 2017.

[60] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[61] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[62] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," 09 2019.

[63] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[64] R. H. Myers and R. H. Myers, *Classical and modern regression with applications*. Duxbury press Belmont, CA, 1990, vol. 2.

[65] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, 2019.

[66] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[67] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[68] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[69] M. Duan, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," *arXiv preprint arXiv:1907.01132*, 2019.

[70] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[71] J. M. Joyce, "Kullback-leibler divergence," *International encyclopedia of statistical science*, pp. 720–722, 2011.

[72] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[73] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in neural information processing systems*, 2014, pp. 3068–3076.

[74] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural, Parallel & Scientific Computations*, vol. 11, no. 4, pp. 351–368, 2003.

[75] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[76] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *arXiv preprint arXiv:1811.12629*, 2018.

[77] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[78] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," *arXiv preprint arXiv:1804.08333*, 2018.

[79] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[80] J. Bloemer, "How to share a secret," *Communications of the Acm*, vol. 22, no. 22, pp. 612–613, 2011.

[81] H. B. Mcmahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *international conference on learning representations*, 2018.

[82] Google, "Tensorflow federated: Machine learning on decentralized data."

[83] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[84] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[85] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, p. 18, 2010.

[86] A. Go, R. Bhayani, and L. Huang, "Sentiment140," *Site Functionality, 2013c. URL http://help. sentiment140. com/site-functionality. Abruf am*, vol. 20, 2016.

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

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

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[90] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning."

[91] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9850–9861.

[92] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

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

[94] Z. Tao and Q. Li, "esgd: Communication efficient distributed deep learning on the edge," in {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[95] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[96] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[97] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4.

[98] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-iid data," *arXiv preprint arXiv:1905.06641*, 2019.

[99] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*. JMLR. org, 2015, pp. 97–105.

[100] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[101] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[102] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3329–3337.

[103] B. S. Kashin, "Diameters of some finite-dimensional sets and classes of smooth functions," *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, vol. 41, no. 2, pp. 334–351, 1977.

[104] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[105] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[106] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[107] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[108] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[109] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[110] M. Buscema and S. Terzi, "Semeion handwritten digit data set," *Center for Machine Learning and Intelligent Systems, California, USA*, 2009.

[111] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 21–28.

[112] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[113] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.

[114] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks," *arXiv preprint arXiv:1905.07210*, 2019.

[115] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L.-C. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *arXiv preprint arXiv:1812.03633*, 2018.

[116] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[117] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions On Networking*, vol. 16, no. 2, pp. 396–409, 2008.

[118] M. Feldman, "Computational fairness: Preventing machine-learned discrimination," 2015.

[119] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[120] D. López-Pérez, A. Valcarce, G. De La Roche, and J. Zhang, "Ofdma femtocells: A roadmap on interference avoidance," *IEEE Communications Magazine*, vol. 47, no. 9, pp. 41–48, 2009.

[121] G. Zhu, Y. Wang, and K. Huang, "Low-latency broadband analog aggregation for federated edge learning," *arXiv preprint arXiv:1812.11494*, 2018.

[122] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, 2011.

[123] M. M. Amiri and D. Gunduz, "Federated learning over wireless fading channels," *arXiv preprint arXiv:1907.09769*, 2019.

[124] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *arXiv preprint arXiv:1812.11750*, 2018.

[125] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[126] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[127] P. D. Tao *et al.*, "The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems," *Annals of operations research*, vol. 133, no. 1-4, pp. 23–46, 2005.

[128] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, "Approximation bounds for quadratic optimization with homogeneous quadratic constraints," *SIAM Journal on optimization*, vol. 18, no. 1, pp. 1–28, 2007.

[129] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[130] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y.-C. Liang, "Joint service pricing and cooperative relay communication for federated learning," *arXiv preprint arXiv:1811.12082*, 2018.

[131] M. J. Osborne *et al.*, *An introduction to game theory*. Oxford university press New York, 2004, vol. 3, no. 3.

[132] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *arXiv preprint arXiv:1908.03092*, 2019.

[133] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," *arXiv preprint arXiv:1905.07479*, 2019.

[134] P. Bolton, M. Dewatripont *et al.*, *Contract theory*. MIT press, 2005.

[135] R. Jurca and B. Faltings, "An incentive compatible reputation mechanism," in *EEE International Conference on E-Commerce, 2003. CEC 2003.* IEEE, 2003, pp. 285–292.

[136] R. Dennis and G. Owen, "Rep on the block: A next generation reputation system based on the blockchain," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015, pp. 131–138.

[137] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning." IEEE, 2019.

[138] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 343–347.

[139] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security*, vol. 10, no. 3, pp. 137–150, 2015.

[140] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[141] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[142] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[143] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[144] P. Laskov *et al.*, "Practical evasion of a learning-based classifier: A case study," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 197–211.

[145] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[146] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[147] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[148] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[149] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 603–618.

[150] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[151] Y. Liu, Z. Ma, S. Ma, S. Nepal, and R. Deng, "Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing," *https://arxiv.org/pdf/1907.10218.pdf*, 2019.

[152] A. Triastcyn and B. Faltings, "Federated generative privacy," *https://pdfs.semanticscholar.org/2bea/364403c24100758a01752bbfa5ba625c296f.pdf*, 2019.

[153] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[154] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preseving federated deep learning," in *2019 IEEE International Conference on Communications*. IEEE, 2019, pp. 1–6.

[155] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[156] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[157] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[158] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *arXiv preprint arXiv:1811.12470*, 2018.

[159] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[160] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," *arXiv preprint arXiv:1808.03949*, 2018.

[161] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *Cryptology ePrint Archive, Report 2018/679*, 2018.

[162] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[163] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014, pp. 241–246.

[164] A. Abeshu and N. Chilamkurti, "Deep learning: the frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[165] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A.-R. Sadeghi, "D\" iot: A crowdsourced self-learning approach for detecting compromised iot devices," *arXiv preprint arXiv:1804.07474*, 2018.

[166] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[167] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

[168] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, "Federated learning based proactive content caching in edge computing,"

in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[169] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Information Sciences*, vol. 505, pp. 562–570, 2019.

[170] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[171] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[172] Y. Saputra, H. Dinh, D. Nguyen, E. Dutkiewicz, M. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[173] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[174] T. U. of New Brunswick, "Nsl-kdd https://www.unb.ca/cic/datasets/nsl.html."

[175] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[176] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[177] S. Priya and D. J. Inman, *Energy harvesting technologies*. Springer, 2009, vol. 21.

[178] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in neural information processing systems*, 2011, pp. 2249–2257.

[179] J. Chung, H.-J. Yoon, and H. J. Gardner, "Analysis of break in presence during game play using a linear mixed model," *ETRI journal*, vol. 32, no. 5, pp. 687–694, 2010.

[180] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046–1061, 2017.

[181] O. Guéant, J.-M. Lasry, and P.-L. Lions, "Mean field games and applications," in *Paris-Princeton lectures on mathematical finance 2010*. Springer, 2011, pp. 205–266.

[182] L. De Haan and A. Ferreira, *Extreme value theory: an introduction*. Springer Science & Business Media, 2007.

[183] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[184] P. You and Z. Yang, "Efficient optimal scheduling of charging station with multiple electric vehicles via v2v," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 716–721.

[185] P. Bradley, K. Bennett, and A. Demiriz, "Constrained k-means clustering," *Microsoft Research, Redmond*, vol. 20, no. 0, p. 0, 2000.

[186] W. Li, T. Logenthiran, V.-T. Phan, and W. L. Woo, "Implemented iot-based self-learning home management system (shms) for singapore," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2212–2219, 2018.

[187] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[188] L. Jiang, R. Tan, X. Lou, and G. Lin, "On lightweight privacy-preserving collaborative learning for internet-of-things objects," 2019.

[189] Z. Gu, H. Jamjoom, D. Su, H. Huang, J. Zhang, T. Ma, D. Pendarakis, and I. Molloy, "Reaching data confidentiality and model accountability on the caltrain," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 336–348.

[190] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3. IEEE, 2000, pp. 2275–2280.

[191] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE network*, vol. 20, no. 3, pp. 41–47, 2006.

[192] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 64–78.

[193] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.

[194] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," *EC*, vol. 99, pp. 129–139, 1999.

[195] I. I. Eliazar and I. M. Sokolov, "Measuring statistical heterogeneity: The pietra index," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 1, pp. 117–125, 2010.

[196] J.-S. Leu, T.-H. Chiang, M.-C. Yu, and K.-W. Su, "Energy efficient clustering scheme for prolonging the lifetime of wireless sensor network with isolated nodes," *IEEE communications letters*, vol. 19, no. 2, pp. 259–262, 2014.