Federated Learning in Mobile Edge Networks: A Comprehensive Survey

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

***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 possibilities 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 and future research directions in FL.**

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

Manuscript received September 26, 2019; revised February 21, 2020; accepted March 21, 2020. Date of publication April 8, 2020; date of current version August 21, 2020. This work was supported in part by the National Research Foundation (NRF), Singapore, through Singapore Energy Market Authority, Energy Resilience, under Grant NRF2017EWT-EP003-041 and Grant NRF2015-NRF-ISF001-2277, in part by the Singapore NRF National Satellite of Excellence, Design Science and Technology for Secure Critical Infrastructure under Grant NSoE DeST-SCI2019-0007, in part by the A*STAR-NTU-SUTD Joint Research Grant Call on Artificial Intelligence for the Future of Manufacturing under Grant RGANS1906 and Grant WASP/NTU M4082187(4080), in part by Singapore MOE Tier 2 under Grant MOE2014-T2-2-015 ARC4/15 and Grant MOE Tier 1 2017-T1-002-007 RG122/17, in part by AI Singapore Programme under Grant AISG-GC-2019-003 and Grant NRF-NRFI05-2019-0002, in part by the Alibaba-NTU Singapore Joint Research Institute under Grant Alibaba-NTU-AIR2019B1, in part by Nanyang Technological University, Singapore, and in part by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.02-2019.305. The work of Ying-Chang Liang was supported in part by the National Natural Science Foundation of China under Grant 61631005 and Grant U1801261, in part by the National Key Research and Development Program of China under Grant 2018YFB1801105, and in part by the 111 Project under Grant B20064. The work of Qiang Yang was supported by Hong Kong CERG under Grant 16209715 and Grant 16244616. *(Corresponding author: Nguyen Cong Luong.)*

Wei Yang Bryan Lim is with the Alibaba Group, Alibaba–NTU Joint Research Institute, Nanyang Technological University, Singapore (e-mail: limw0201@e.ntu.edu.sg).

Nguyen Cong Luong is with the Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam, and also with the Phenikaa Research and Technology Institute, A&A Green Phoenix Group JSC, Hanoi 11313, Vietnam (e-mail: luong.nguyencong@phenikaa-uni.edu.vn).

Dinh Thai Hoang is with the School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: hoang.dinh@uts.edu.au).

Yutao Jiao and Dusit Niyato are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: yjiao001@e.ntu.edu.sg; dniyato@ntu.edu.sg).

Ying-Chang Liang is with the Center for Intelligent Networking and Communications, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: liangyc@ieee.org).

Qiang Yang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: qyang@cse.ust.hk).

Chunyan Miao is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, also with the Alibaba–NTU Joint Research Institute, Nanyang Technological University, Singapore, and also with the Joint NTU–UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore (e-mail: ascymiao@ntu.edu.sg).

Digital Object Identifier 10.1109/COMST.2020.2986024

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 U.S. [11]. In particular, the consent (GDPR Article 6) and data minimalization principle (GDPR Article 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 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 more 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 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. This discourages the development of technologies and 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. Compared to conventional cloud-centric training approaches, the implementation of FL for model training at mobile edge networks features the following advantages.

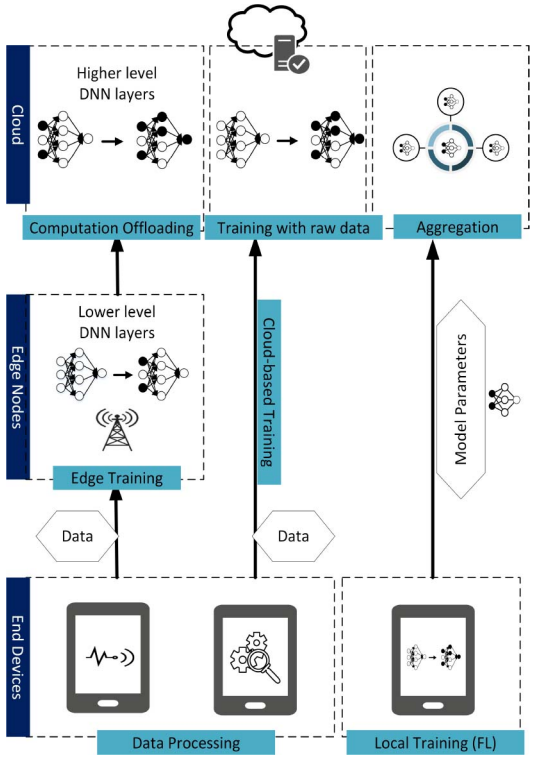


Fig. 1. Edge AI approach brings AI processing closer to where data is produced.

- *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. Under the assumption that FL participants and servers are non-malicious, this enhances user privacy and reduces the probability of eavesdropping to a certain extent. In fact, with enhanced privacy, more users will be willing to take part in collaborative model training and so, better inference models can be 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 hospitals [26] and Government agencies [27].

Besides being an enabling technology for ML model training *at* mobile edge networks, FL has also been increasingly applied as 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 [28] for optimizing resource allocation is increasingly popular. For example, DL can be used for representation learning of network conditions [29] whereas Deep Reinforcement Learning (DRL) can optimize decision making through interactions with the dynamic environment [30]. 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 [31], computation offloading [32], and vehicular networks [33].

However, there are several challenges to be solved before FL can be implemented at scale. Firstly, even though raw data no longer needs to be sent to the cloud servers, communication costs remain an issue due to the high dimensonality of model updates and limited communication bandwith of participating mobile devices. In particular, state-of-the art DNN model training can involve the communication of millions of parameters for aggregation. 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, FL does not guarantee privacy in the presence of malicious participants or servers. In particular, recent research works have clearly shown that a malicious participant may exist in FL and can infer the information of other participants just from the shared parameters alone. As such, privacy and security issues in FL still 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 [34] 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 [35] highlight the applications of FL in wireless communications but do not discuss the issues pertaining to FL implementation. In addition, the focus of [35] is on cellular network architecture rather than mobile edge networks. In contrast, the authors in [36] 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], [37], and [38] 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 [12], [19], [39]–[42] discuss strategies for optimizing caching and computation offloading for mobile edge networks, but do not consider the use of privacy preserving federated approaches in their studies. Similarly, [30] 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 applications of FL in the context of mobile edge networks, whereas surveys on MEC do not consider the challenges to FL implementation, or the potential of FL to be applied in mobile edge network optimization. This motivates us to have a comprehensive survey with the following contributions:

- We motivate the importance of FL as an important paradigm shift towards enabling collaborative ML model training. Then, we provide a concise tutorial on FL implementation and present to the reader a list of useful open-source frameworks that paves the way for future research on FL and its applications.
- We discuss the unique features of FL relative to a centralized ML approach and the resulting implementation challenges. For each of this challenge, we present to the reader a comprehensive discussion of existing solutions and approaches explored in the FL literature.
- We discuss FL as an enabling technology for mobile edge network optimization. In particular, we discuss the current and potential applications of FL as a privacy-preserving approach for applications in edge computing.
- We discuss challenges and research directions of FL.

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 solving the challenges and issues related to implementing the collaborative training of ML models on end devices, and (ii) FL for mobile edge network, i.e., studies that specifically explore the application of FL for mobile edge network optimization. While the former group of studies works on addressing the fundamental issues of FL, the latter group uses FL as an application tool to solve issues in edge computing. We also present a list of common abbreviations for reference in Table II.

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 and future research directions in FL. Section VIII concludes the paper.

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) [43]

TABLE I
AN OVERVIEW OF SELECTED SURVEYS IN FL AND MEC

Ref.	Subject	Contribution
[34]	FL	Introductory tutorial on categorization of different FL settings, e.g., vertical FL, horizontal FL, and Federated Transfer Learning
[35]		FL in optimizing resource allocation for wireless networks while preserving data privacy
[36]		Tutorial on FL and discussions of implementation challenges in FL
[14]	MEC	Computation offloading strategy to optimize DL performance in edge computing
[37]		Survey on architectures and frameworks for edge intelligence
[38]		ML for IoT management, e.g., network management and security
[19]		Survey on computation offloading in MEC
[30]		Survey on DRL approaches to address issues in communications and networking
[39]		Survey on techniques for computation offloading
[40]		Survey on architectures and applications of MEC
[41]		Survey on computing, caching, and communication issues at mobile edge networks
[42]		Survey on the phases of caching and comparison among the different caching schemes
[12]		Survey on joint mobile computing and wireless communication resource management in MEC

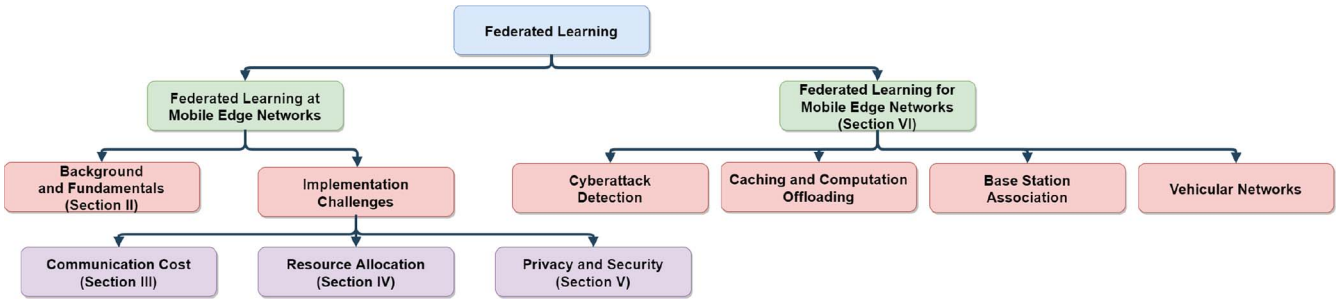


Fig. 2. Classification of related studies to be discussed in this survey.

TABLE II
LIST OF COMMON ABBREVIATIONS

Abbreviation	Description
BAA	Broadband Analog Aggregation
CNN	Convolutional Neural Network
DDQN	Double Deep Q-Network
DL	Deep Learning
DNN	Deep Neural Network
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
FedAvg	Federated Averaging
FL	Federated Learning
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
QoE	Quality of Experience
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SNR	Signal-to-noise ratio
SVM	Support Vector Machine
TFF	TensorFlow Federated
UE	User Equipment

and Natural Language Processing (NLP) [44]. 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 on end devices and (ii) global aggregation of updated parameters in the FL server. 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 as a majority of the papers that we subsequently review study the federated training of DNN models. In addition, the DNN models are easily aggregated and outperform conventional ML techniques especially when the data is large. The implementation of FL at mobile edge networks can thus 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 [45]. 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 [46]. 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. Otherwise, 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. The neural network comprises three layers: (i) input layer, (ii) hidden layer, and (iii) output layer. In a conventional DNN, a weighted and bias-corrected input value is passed through a non-linear activation function to derive an output. Some activation functions include the ReLu and softmax functions [44]. 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 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 is minimized.

Before training, the dataset is first split into the training and test 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., the step size of gradient descent in each iteration, and (ii) partial derivative of the loss function *L* with respect to the weight *w*. The SGD formula is as follows:

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

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

Note that the SGD formula presented in (1) is that of a mini-batch GD. Equation (2) is derived as the average gradient matrix over the gradient matrices of *B* batches, in which each batch is a random subset including *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 [47]. The gradient matrices are derived through back propagation from the input gradient.

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 [48], unsupervised learning [49] and reinforcement learning [50]. In addition, there also exists several DNN networks and architectures tailored to process the varying natures of input data, e.g., Multilayer Perceptron (MLP) [51], Convolutional Neural Network (CNN) [52] typically for CV tasks, and Recurrent Neural Network (RNN) [53] usually for sequential tasks. However, an in-depth discussion is out of the scope of this paper. We refer interested readers to [54]–[59] for comprehensive discussions of DNN architectures and training strategies. We next focus on FL, an important paradigm shift towards enabling privacy preserving and collaborative DL model training.

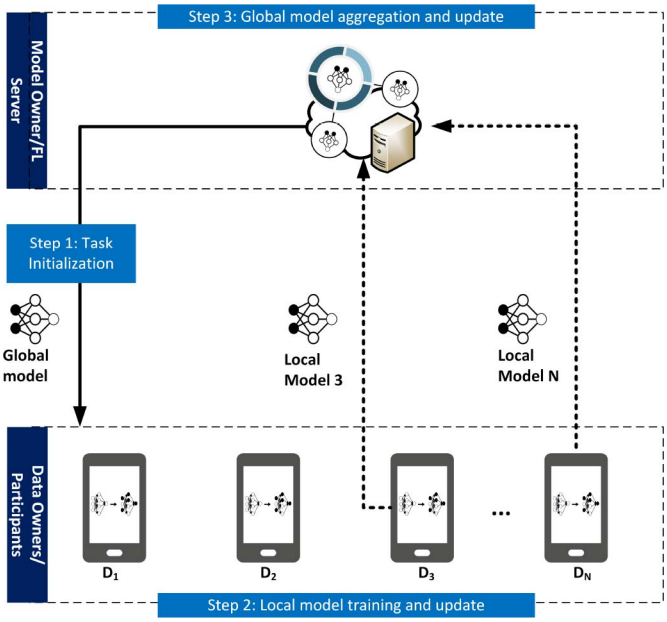


Fig. 3. General FL training process involving *N* participants.

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. For an introduction to the categorizations of different FL settings, e.g., vertical and horizontal FL, we refer the interested readers to [34].

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 first before model training takes place centrally.

A typical architecture and training process of an FL system is shown in Fig. 3. 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 [60] 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

TABLE III
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.)

data requirements. The server also specifies the hyper parameters 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.,

$$\mathbf{w}_i^{t^*} = \arg\min_{\mathbf{w}_i^t} L\big(\mathbf{w}_i^t\big). \tag{3}$$

  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 minimizes the global loss function $L(\mathbf{w}_G^t)$, i.e.,

$$L\big(\mathbf{w}_G^t\big) = \frac{1}{N}\sum_{i=1}^{N} L\big(\mathbf{w}_i^t\big). \tag{4}$$

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) [61], neural networks, and linear regression [62]. 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 III summarizes several loss functions of common ML models [63].

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 similar to that of local SGD [64]. The pseudocode for *FedAvg* is 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 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.

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*, **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**

C. Statistical Challenges of 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 [65] found otherwise. For example, the accuracy of a *FedAvg*-trained CNN model has 51% lower accuracy than centrally-trained CNN model for CIFAR-10 [66]. This deterioration in accuracy is further shown to be quantified by the earth mover's distance (EMD) [67], 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

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

FL server. An alternative solution is to gather contributions towards building the common dataset.

The authors in [68] also find that global imbalance, i.e., the situation in which the collection of data held across all FL participants is class imbalanced, leads to a deterioration in model accuracy. As such, the Astraea framework is proposed. On initialization, the FL participants send their data distribution to the server. A rebalancing step is introduced before training begins in which each participant performs data augmentation [69] 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 [70] between local data and uniform distribution. The simulation results show accuracy improvement when tested on imbalanced datasets.

Given the heterogeneity of data distribution across devices, there has been an increasing number of studies that borrow concepts from multi-task learning [71] to learn separate, but structurally related models for each participant. Instead of minimizing the conventional loss function presented previously in Table III, the loss function is modified to model the relationship amongst tasks. Then, the MOCHA algorithm is proposed in which an alternating optimization approach [72] 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 devices. However, MOCHA cannot be applied to non-convex DL models.

Similarly, the authors in [73] borrow concepts from multi-task learning to deal with the statistical heterogeneity in FL. The FEDPER approach is proposed in which FL participants share a set of base layers trained using the *FedAvg* algorithm. Then, each participant separately trains another set of personalization layers using its local data. In particular, this approach is suitable for building recommender's systems given the diverse preferences of participants. The authors show empirically using the Flickr-AES dataset [74] that the FEDPER approach outperforms a pure *FedAvg* approach since the personalization layer is able to represent the personal preference of an FL participant. However, it is worth noting that the collaborative training of the base layers are important to achieve a high test accuracy since each participant has insufficient local data samples for purely personalized model training.

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 imply reduced 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 prevailing 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 [65] 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 to solve 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]. This protocol deals with issues regarding unstable device connectivity and communication security etc. The FL protocol (Fig. 4) consists of three phases in each training round:

1) *Selection:* 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 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.

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:

- *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.

- *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:

- *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.
- *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.
- *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 devices. The implementation of newly designed algorithms on these datasets allow for reliable comparison across studies.
- *FATE:* Federated AI Technology Enabler (FATE) is an open-source framework by WeBank [87] that supports the federated and secure implementation of ML models.

E. Unique Characteristics and Issues of FL

FL has some unique characteristics and features [88] as compared to other distributed ML approaches.

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 [89]. Also, some participants with unstable wireless communication channels may consequently drop out due to disconnection from the Internet.
2) *Heterogeneous devices:* 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, i.e., (i) statistical challenges (ii) communication costs (iii) resource allocation and (iv) privacy and security. The following sections 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. 4). For complex DL model training involving, e.g., CNN, each update may comprise millions of parameters [90]. 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 [91] 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 [89]. 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 the model training only if they are connected to Wi-Fi [89]. As such, more computation can be performed on edge nodes or on end devices before each global aggregation so as to reduce the number

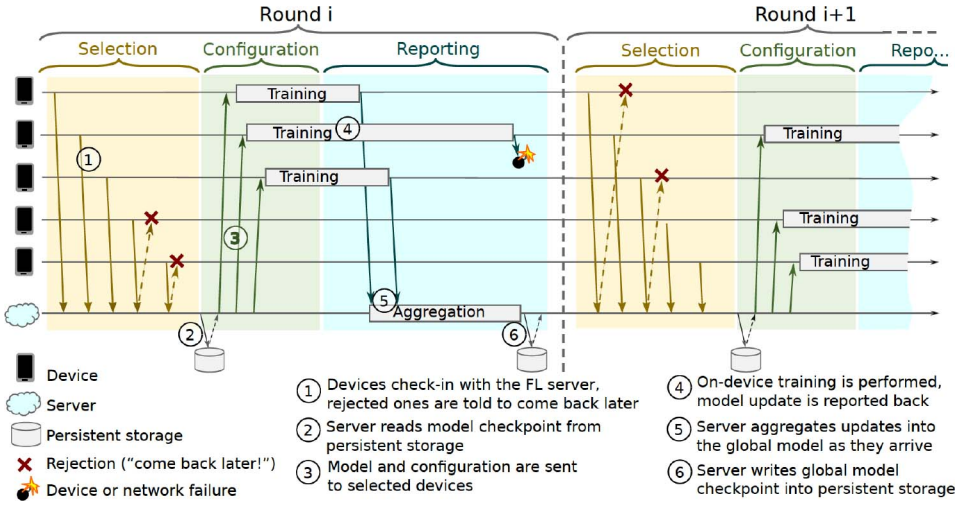


Fig. 4. Federated learning protocol [77].

of communication rounds needed for the model training. In addition, algorithms to ensure faster convergence can reduce number of 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 [92]. Model or gradient compression involves the communication of an update that is transformed to be more compact, e.g., through sparsification, quantization or subsampling [93], rather than the communication of full update. However, since the compression may introduce noise, the objective is to reduce the size of update transferred during each round while maintaining the quality of trained models [94].
- *Importance-based Updating:* This strategy involves selective communication such that only the important or relevant updates [95] are transmitted in each communication round. In fact, besides saving on communication costs, omitting some updates from participants can even improve the global model performance.

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 communication for global aggregation (Fig. 5(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 hyper parameters are tuned such that more local computations are 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 [96], 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 [97], which prevents overfitting.

As an extension, the authors in [98] also validate that a similar concept as that of [23] works for vertical FL. In vertical FL, collaborative model training is conducted across the same set of participants with different data features. The Federated Stochastic Block Coordinate Descent (FedBCD) algorithm is proposed in which each participating device performs multiple local updates first before communication for global aggregation. In addition, convergence guarantee is also provided with an approximate calibration of the number of local updates per interval of communication.

Another 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 [99] also propose increased computation on each participating device by adopting a two-stream model (Fig. 5(b)) commonly used in transfer learning and domain adaption [101]. 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

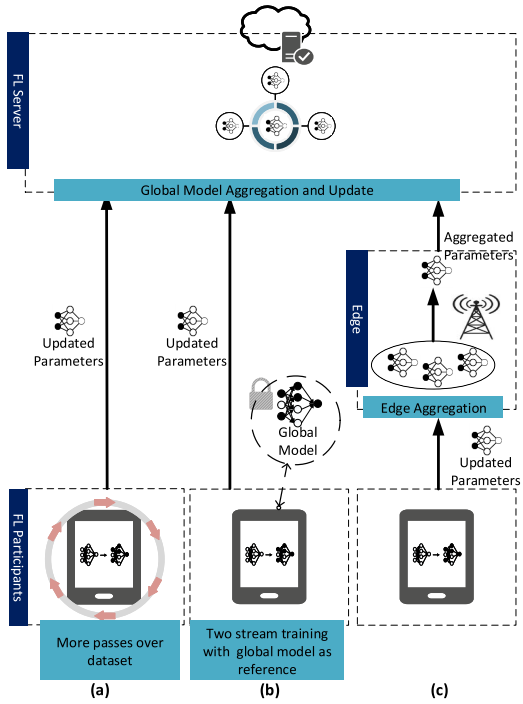


Fig. 5. Approaches to increase computation at edge and end devices include (a) Increased computation at end devices, e.g., more passes over dataset before communication [23], [98] (b) Two-stream training with global model as a reference [99] and (c) Intermediate edge server aggregation [100].

through the incorporation of Maximum Mean Discrepancy (MMD) into the loss function. MMD measures the distance between the means of two data distributions [101], [102]. 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 [52] 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. Given the energy constraints of participating mobile devices in particular, 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 [100] propose an edge computing inspired paradigm in which 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-cloud communication (Fig. 5(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 covergence 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 [89] propose structured and sketched updates to reduce the size of model updates sent from participants to the FL 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 is the probabilistic quantization approach [103], 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 [104] is applied before quantization.

The simulation results on the CIFAR-10 image classification task show that for structured updates, the random

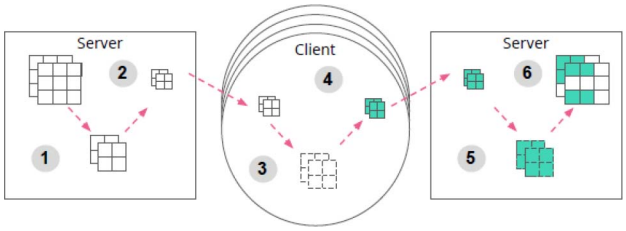


Fig. 6. The compression techniques considered are summarized above by the diagram from authors in [94]. (i) Federated dropout to reduce size of model (ii) Lossy compression of model (iii) Decompression for training (iv) Compression of participant updates (v) Decompression (vi) Global aggregation.

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.

The authors in [94] extend the studies in [89] 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. 6. For participant-to-server upload 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 [89], subsampling and probabilistic quantization are considered. For the application of structured random rotation before subsampling and quantization, Kashin's representation [105] is applied instead of the Hadamard transformation since the former is found to perform better in terms of accuracy-size tradeoff.

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 [106] datasets. For the lossy compression, it is shown that the subsampling approach taken by [89] 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 parameter values of a DNN model are sparsely distributed and close to zero [107], the authors in [95] 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 values. 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 [108], 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 [109]. 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 threshold SGD algorithm proposed in [107], 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.

While [95] studies the selective communication of gradients, the authors in [91] propose the Communication-Mitigated Federated Learning (CMFL) algorithm that uploads only relevant local model 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 [110] which measures relevance based on *magnitude* of updates rather than sign of parameters. When applied with the aforementioned MOCHA algorithm Section II-C, CMFL can reduce communication rounds by 5.7 times for the Human Activity Recognition dataset [111] and 3.3 times for the Semeion Handwritten Digit dataset [112]. In addition, CMFL achieves slightly higher accuracy since it involves the elimination of irrelevant updates that are outliers which harm training.

D. Summary and Lessons Learned

In this section, we have reviewed three main approaches for communication cost reduction in FL. We summarize the approaches along with references in Table IV. From this review, we gather the following lessons learned:

- Communication cost is a key issue to be resolved before we can implement FL at scale. In particular, the state-of-the-art DL models have high inference accuracy but are increasingly complex with millions of parameters. The slow upload speed of mobile devices can thus impede the implementation of efficient FL.
- This section explores several key approaches to communication cost reduction. However, many of the approaches, e.g., model compression, result in a deterioration in model accuracy or incur high computation cost. For example, when too many local updates are implemented between communication rounds, the communication cost is indeed reduced but the convergence can be significantly delayed [98]. The tradeoff between these sacrifices and communication cost reduction thus has to be well-managed.
- The current studies of this tradeoff are often mainly empirical in nature, e.g., several experiments have to be done to find the optimal number of local training iterations before communication. With more effective optimization approaches formalized theoretically and tested empirically, FL can eventually become more scalable in nature. For example, the authors in [113] study the tradeoffs between the completion time of FL training and energy cost expended. Then, a weighted sum of completion time and energy consumption is minimized using an iterative algorithm. For delay-sensitive scenarios, the weights can be adjusted such that the FL participants consume more energy for completion time minimization.
- Apart from working to directly reduce the size of model communicated, studies on FL can draw inspiration from applications and approaches in the MEC paradigm. For example, a simple case study introduced in [95] considers the base station as an intermediate model aggregator to reduce instances of device-cloud communication. Unfortunately, there are convergence issues when more edge servers or mobile devices are considered. This is exacerbated by the non-IID distribution of data across different edge nodes. For future works, this statistical challenge can be met, e.g., through inspirations from multi-task learning as we have discussed in Section II-C. In addition, more effective and innovative system models can be explored such that FL networks can utilize the wealth of computing and storage resources that are closer to the data sources to facilitate efficient FL.
- For the studies that we have discussed in this section, the heterogeneity among mobile devices, e.g., in computing capabilities, is often not considered. For example, one of the ways to reduce communication cost is to increase computation on edge devices, e.g., by performing more local updates [23] before each communication round. In fact, this does not merely lead to the expenditure of greater computation cost. The approach may also not be feasible for devices with weak processing power, and can lead to the *straggler effect*. As such, we further explore issues on resource allocation in the next 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.

TABLE IV
APPROACHES TO COMMUNICATION COST REDUCTION IN FL

Approaches	Ref.	Key Ideas	Tradeoffs and Shortcomings
Edge and End Computation	[23]	More local updates in between communication for global aggregation, to reduce instances of communication	Increased computation cost and poor performance in non-IID setting
	[98]	Similar to the ideas of [23], but with convergence guarantees for vertical FL	Increased computation cost and delayed convergence
	[99]	Transfer learning-inspired two-stream model for FL participants to learn from the fixed global model to accelerate training convergence	Increased computation cost and delayed convergence
	[100]	MEC-inspired edge server assisted FL that aids in intermediate parameter aggregation to reduce instances of communication	System model is not scalable when there are more edge servers
Model Compression	[89]	Structured and sketched updates to compress local models communicated from participant to FL server	Model accuracy and convergence issues
	[94]	Similar [89], but for communication from FL server to participants	Model accuracy and convergence issues
Importance-based Updating	[95]	Selective communication of gradients that are assigned importance scores, i.e., to reduce training loss	Only empirically tested on simple datasets and tasks, with fluctuating results
	[91]	Selective communication of local model updates that have higher relevance scores when compared to previous global model	Difficult to implement when global aggregations are less frequent

- *Participant Selection:* 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 participants 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 [114]. New participant selection protocols are thus investigated to address the training bottleneck in FL.
- *Joint Radio and Computation Resource Management:* Even though computation capabilities of mobile devices have grown rapidly, many devices still face a scarcity of radio resources [115]. Given that local model transmission is an integral part of FL, there has been a growing number of studies that focus on developing novel wireless communication techniques for efficient FL.
- *Adaptive Aggregation:* 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., global aggregations occur in fixed intervals after all participants complete a certain number of rounds of local computation. However, adaptive calibrations of global aggregation frequency can be investigated to increase training efficiency subject to resource constraints [114].
- *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. 7. 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 [116] 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. For 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 [117] subsequently in this section.

While *FedCS* addresses heterogeneity of resources among participants in FL, the authors in [118] 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 [65]. 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.

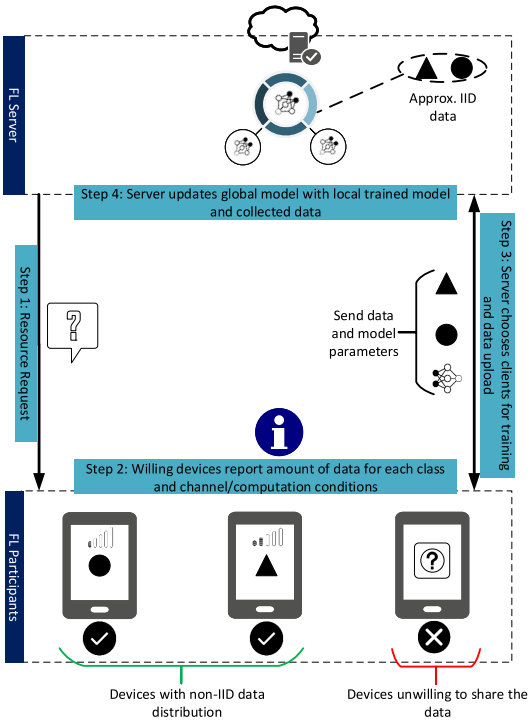


Fig. 7. Participant selection under FedCS and Hybrid-FL protocols.

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 using inputs from a limited number of privacy insensitive participants [118]. In the Hybrid-FL protocol, during the *Resource Request* step (Fig. 7), 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. 7). 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. 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. To this end, Deep Q-Learning (DQL) can be used to optimize resource allocation for model training as proposed in [119]. The system model includes participants, i.e., mobile devices, that collaboratively train DNN models required by a FL server. The 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. A stochastic optimization problem is formulated in which 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. The reward is defined as a function of the accumulated data, energy consumption, and training latency. The Double Deep Q-Network (DDQN) [120] is then adopted to solve the server's problem. The simulation results show that the proposed 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.

As an extension to [119], the authors in [121] propose a resource allocation approach using DRL, with the added uncertainty that FL participants are mobile and so they may venture out of the network coverage range. Without prior knowledge of the mobile network, the FL server is able to optimize resource allocation across participants, e.g., channel selection and device energy consumption.

The aforementioned resource allocation approaches focus on improving the training efficiency of FL. However, this may cause some FL participants to be left out of the aggregation phase because they are stragglers with limited resources.

One consequence of this *unfair* resource allocation, a topic that is commonly explored in resource allocation for wireless networks [122] and ML [123]. 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 over represented by the distribution of data owned by participants with devices that have higher computing capabilities. Therefore, the authors in [117] and [124] consider fairness as an additional objective in FL. Fairness is defined in [124] 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 under represented participants. The authors in [124] 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. The modified objective function is as follows:

$$\min_{w} F_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w) \tag{6}$$

where F_k refers to the standard loss functions presented in Table III, q refers to the calibration of fairness in the system model, i.e., setting $q = 0$ returns the formulation to the typical FL objective, and p_k refers to ratio of local samples to the total number of training samples. In fact, this is a generalization of the Agnostic FL (AFL) algorithm proposed in [117], 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 can be considered for use with the q-FFL algorithm.

In contrast, the authors in [125] propose a neural network based approach to estimate the local models of FL participants that are left out during training. In the system model, resource blocks are first allocated by the base station to users whose models have larger effects on the global FL model. In particular, one user is selected to always be connected to the base station. This user's model parameters are then used as input to the feed forward neural network to estimate the model parameters of users who are left out during the training iteration. This allows the base stations to be able to integrate more locally trained FL model parameters to each iteration of global aggregation, thus improving the FL convergence speed.

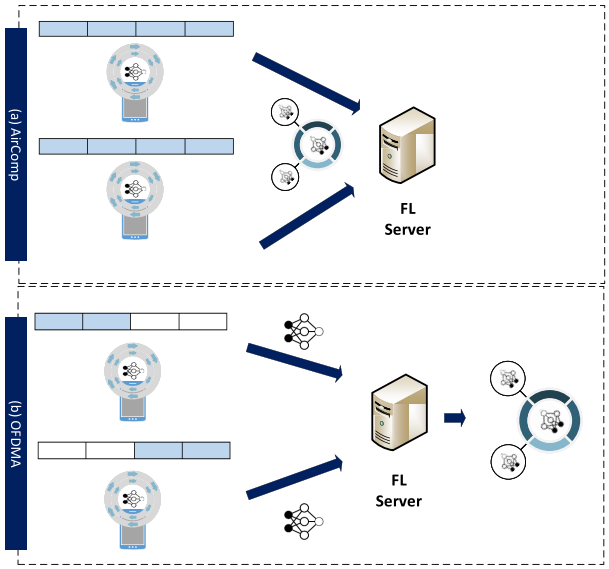


Fig. 8. A comparison [127] between (a) BAA by over-the-air computation which reuses bandwidth (above) and (b) OFDMA (below) which uses only the allocated bandwidth.

B. Joint Radio and Computation Resource Management

While most FL studies have previously assumed orthogonal-access schemes such as Orthogonal Frequency-division Multiple Access (OFDMA) [126], the authors in [127] 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 [128] 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. 8(a)) whereas OFDMA orthogonalizes bandwidth allocation (Fig. 8(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 [129] introduce error accumulation and gradient sparsification in addition to over-the-air computation. In [127], gradient vectors that are not transmitted as a result of power constraints are completely dropped. To improve the model accuracy, the untransmitted gradient vectors are first 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 [129], the authors in [130] propose an integration of computation and communication via over-the-air computation. However, it is observed that aggregation error incurred during over-the-air computation can lead to a drop in model accuracy [131] 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 to improve statistical learning performance [23] while keeping the signal distortion below a threshold. Due to the nonconvexity of the mean-square-error constraint and intractability of the optimization problem, a difference-of-convex functions (DC) algorithm [132] is proposed to solve the maximization problem. The simulation results show that the proposed algorithm is scalable and can also achieve near-optimal performance that is comparable to global optimization,

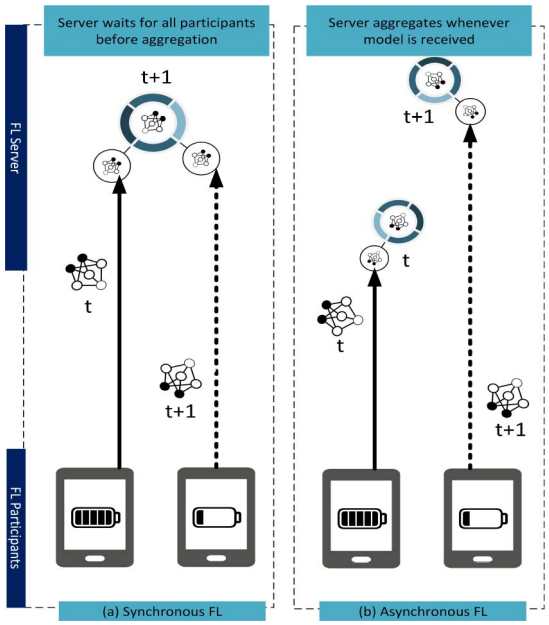


Fig. 9. A comparison between (a) synchronous and (b) asynchronous FL.

which is non-scalable due to its exponential time complexity. In comparison with other state-of-the-art approaches such as the semi definite relaxation technique [133], the proposed algorithm can select more participants, thus achieving higher accuracy.

C. Adaptive Aggregation

The proposed *FedAvg* algorithm synchronously aggregates parameters as shown in Fig. 9(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 [114]. 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. 9(b)). The authors in [114] find empirically that an asynchronous approach is robust to participants joining halfway during a training round, as well as when the federation 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, [134] 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 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 [63] 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 [63], 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.

D. Incentive Mechanism

The authors in [135] 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 [136] 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

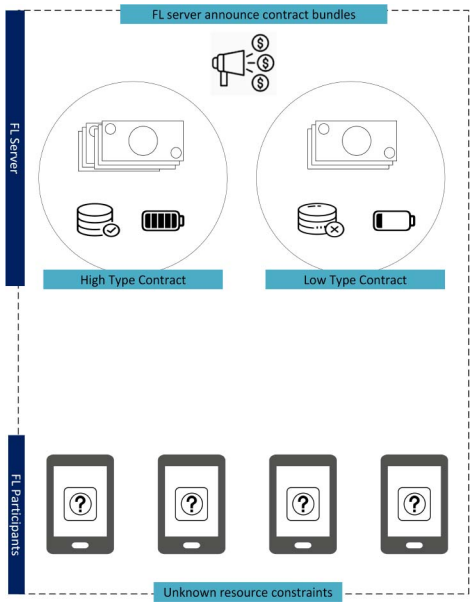


Fig. 10. Participants with resource constraints maximize their utility only if they choose the bundle that best reflects their constraints.

simulation environment only involves relatively few mobile devices.

Similar to [135], the authors in [137] model the interaction between participants and model owner as a Stackelberg game, which is well-suited to represent the FL server-participant interaction involved in FL.

Unlike the aforementioned approaches to solving Stackelberg formulations, a DRL-based approach is adopted together with the Stackelberg game as proposed in [138]. In the DRL formulation, the FL server acts as an agent that decides a payment in response to the participation level and payment history of edge nodes, with the objective of minimizing incentive expenses. Then, the edge nodes determine an optimal participation level in response to the payment policy. This learning based incentive mechanism enables the FL server to derive an optimal policy in response to its observed state, without requiring any prior information.

In contrast to [135], [137]–[139], the authors in [140] propose an incentive design using a contract theoretic [141] 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. 10, 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 contract approach has better performance in terms of profit for the model owner compared with the Stackelberg game approach. 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. In fact, the information asymmetry between FL servers and participants make contract theory a powerful and efficient tool for mechanism design in FL. As an extension, the authors in [142] introduced a *multi-dimensional* contract in which each FL participant determines the optimal computation power and image quality it is willing to contribute for model training, in exchange for contract rewards in each iteration.

The authors in [140] further introduce reputation as a metric to measure the reliability of FL participants and design a reputation-based participant selection scheme for reliable FL [60]. In this setting, each participant has a reputation value 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 [143] 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 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.

E. Summary and Lessons Learned

In this section, we have discussed four main issues in resource allocation. The approaches are summarized in Table V, and the lessons learned are as follows:

- In heterogeneous mobile networks, the consideration of resource allocation is important to ensure efficient FL. For example, each training iteration is only conducted as quickly as the slowest FL participant, i.e., the straggler effect. In addition, the model accuracy is highly dependent on the quality of data used for training by FL participants. In this section, we have explored different dimensions of resource heterogeneity for consideration, e.g., varying computation and communication capabilities, willingness to participate, and quality of data for local model training. In addition, we have explored various tools that can be considered for resource allocation. For example, DRL is useful given the dynamic and uncertain wireless network conditions experienced by FL participants, whereas contract theory can serve as a powerful tool in mechanism design under the context of information asymmetry. Naturally, traditional

TABLE V
APPROACHES TO RESOURCE ALLOCATION IN FL

Approaches	Ref.	Key Ideas	Tradeoffs and Shortcomings
Participant Selection	[78]	FedCS selects participants based on computation capabilities to complete FL training before deadline	Difficult to estimate training duration accurately for complex models
	[118]	Following [78], Hybrid-FL selects participants to accumulate IID, distributable data for FL model training	Request of data sharing may defeat the original intent of FL
	[119]	DRL to determine resource consumption by FL participants	DRL models are difficult to train with the large number of participants
	[121]	Following [119], DRL for resource allocation with mobility-aware FL participants	
	[124]	Fair resource allocation to reduce variance of model performance	Convergence delays with more fairness
Joint Radio and Computation Resource Management	[127]	Integrate computation and communication through exploiting the signal superposition property of multiple-access channel	Signal distortion can lead to drop in accuracy, the scalability is also an issue when large heterogeneous networks are involved
	[129]	Improves on [127] by accounting for gradient vectors that are not transmitted due to power constraints	
	[130]	Improves on [127] using the DC algorithm to minimize aggregation error	
Adaptive Aggregation	[114]	Asynchronous FL where model aggregation occurs whenever local updates are received by FL server	Significant delay in convergence in non-IID and unbalanced dataset
	[63]	Adaptive global aggregation based on resource constraints	Convergence guarantees are limited to restrictive assumptions
Incentive Mechanism	[135], [137]–[139]	Stackelberg game for incentivizing higher quantities of training data or compute resource contributed	FL server derives lower profits. Only one FL server is considered
	[140], [142]	Contract theoretic approach to incentivize FL participants	Only one FL server is considered
	[60]	Reputation mechanism to select effective workers	

optimization approaches have also been well explored in radio resource management for FL, given the high dependency on communications efficiency in FL.

- In Section III, communication cost reduction comes with a sacrifice in terms of either higher computation costs or lower inference accuracy. Similarly, there exist different tradeoffs to be considered in resource allocation. A scalable model is thus one that enables customization to suit varying needs. For example, the study of [124] allows the FL server to calibrate levels of fairness when allocating training importance, whereas the study in [113] enables the tradeoffs between training completion time and energy expense to be calibrated by the FL system.
- In synchronous FL, the FL system is susceptible to the straggler effect. As such, asynchronous FL has been proposed as a solution in [114] and [134]. In addition, asynchronous FL 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 advantages of asynchronous FL, new asynchronous algorithms should be investigated. In particular, for future proposed algorithms, the convergence guarantee in a non-IID setting for non-convex loss functions needs to be considered.
- The study of incentive mechanism design is a particularly important aspect of FL. In particular, due to data privacy concerns, the FL servers are unable to check for training data quality. With the use of self-revealing mechanisms in contract theory, or through modeling the interactions between FL server and participants with game theoretic concepts, high quality data can be motivated as contributions from FL participants. However, existing studies in [60], [135], and [140] generally assume that a federation enjoys a monopoly. In particular, each system model is assumed to only consist of multiple individual participants collaborating with a sole FL server. There can be exceptions to this setting as follows: (i) the participants may be competing data owners who are reluctant to share their model parameters since the competitors also benefit from a trained global model and (ii) the FL servers may compete 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. A relatively novel approach has been to model the regret [144] of each FL participants in joining the various competing federations for model training. For future works, a system model with multiple competing federations can be considered together with Stackelberg games and contract theoretic approaches.
- In this section, we have assumed that FL assures the privacy and security of participants. However, as discussed in the following section, this assumption may not hold in the presence of malicious participants or FL server.

V. PRIVACY AND SECURITY 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 privacy and security concerns may arise when the FL participants or FL servers are malicious in nature. In particular, this defeats the purpose of FL since the resulting global model can be corrupted, or the participants may even have their privacy compromised during model training. In this section, we discuss the following issues:

- *Privacy:* Even though FL does not require the exchange of data for collaborative model training, 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 [145], when training a binary gender classifier on the FaceScrub [146] 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 preserve the privacy of participants.

- *Security:* 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.

A. Privacy Issues

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 [147]. 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 [148], 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 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 [149] 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 [150]–[152]. 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 [153], 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 [154] 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 each training 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 [155] 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. 11. 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 [156], 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

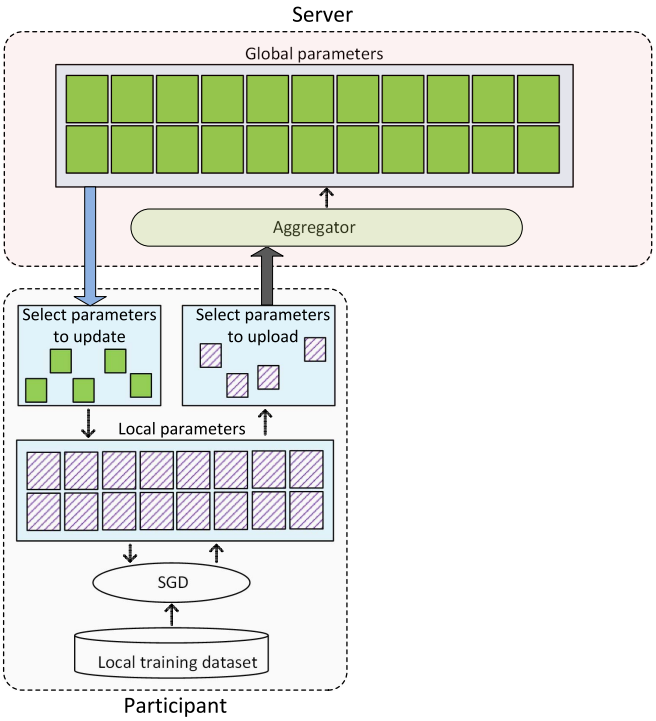


Fig. 11. Selective parameters sharing model.

authors in [157] show that these solutions are susceptible to a new type of attack, called powerful attack, developed based on Generative Adversarial Networks (GANs) [158]. GANs is a class of ML technique which uses two neural networks, 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 [157] 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. 12. To deal with the GAN attack, the authors in [159] 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 [160] 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, 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 [161], 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 [161] 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 [162] 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, the authors do not compare the accuracy of the proposed approach with the case without homomorphic encryption and DP.

B. Security Issues

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 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 [163]. This calls for urgent solutions to deal with data poisoning attacks in FL.

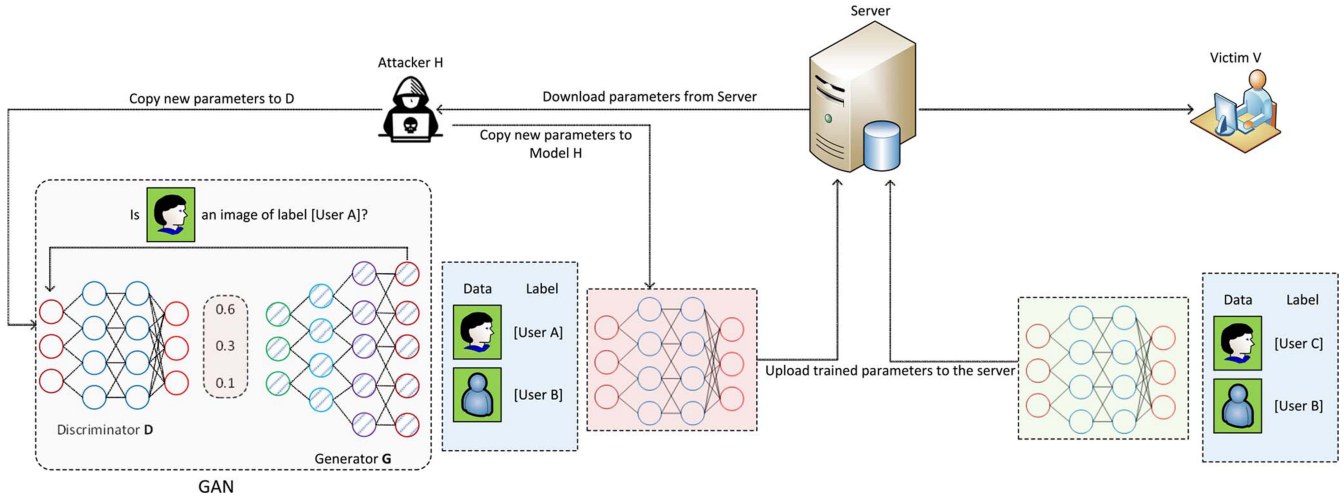


Fig. 12. GAN Attack on collaborative deep learning.

TABLE VI
THE ACCURACY AND ATTACK SUCCESS RATES FOR NO-ATTACK SCENARIO AND ATTACKS WITH 1 AND 2 SYBILS IN A FL SYSTEM WITH MNIST DATASET [156]

	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%

In [164], 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 VI, 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 [156], KDDCup [165], Amazon Reviews [165]), 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 [166] and [167], model poisoning 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 [166] 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 [166], 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 to evaluate the improvement from every single participant. As such, more effective solutions need to be further investigated.

In [167], 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,

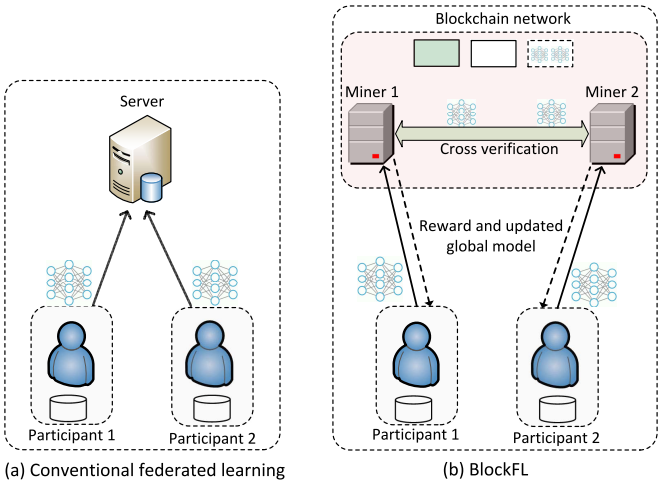


Fig. 13. An illustration of (a) conventional FL and (b) the proposed BlockFL architectures.

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 misclassified to be cats. The simulations results show that this attack can greatly outperform 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 [168] 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. 13. 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 [169] 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.

C. Summary and Lessons Learned

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 have shown 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 the appropriate countermeasures. We summarize the key information of attacks and their corresponding countermeasures in Table VII.

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 discussed in [35], the increasing complexity and heterogeneity of wireless networks enhance the appeal of adopting a data-driven ML based approach [28] 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 [170] imply that there is a need to improve existing cyberattack detection tools. Recently, DL has been widely successful in cyberattack detection. Coupled with FL, cyberattack detection models can be learned collaboratively while maintaining 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 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

TABLE VII
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 [154]. • *Selective parameter sharing*: Each participant wisely selects the number of gradients to upload and the number of parameters from the global model to update [155]. • *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 [159]. • *GAN model training*: All participants are cooperative to train a federated GANs model [160].
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 [164].
Model poisoning attacks	Attackers attempt to 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 [166]. • 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 [166].
Free-riding attacks	Attackers benefit from the global model without contributing to the learning process.	• *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 [168].

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.

- *Vehicular Networks:* The Internet of Vehicles (IoV) [171] 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 [181], 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 [182], and UNSW-NB15 [183]. 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 [172], 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. 14. 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 [173] 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 [173] 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 [172] and [173], 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 [174] propose to use blockchain technology in managing data shared by the participants. By using the 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 [32] propose the use of DRL with FL to optimize caching and computation offloading

TABLE VIII
FL BASED APPROACHES FOR MOBILE EDGE NETWORK OPTIMIZATION

Applications	Ref.	Description
Cyberattack Detection	[172]	Cyberattack detection with edge nodes as participants
	[173]	Cyberattack detection with IoT gateways as participants
	[174]	Blockchain to store model updates
Edge caching and computation offloading	[32]	DRL for caching and offloading in UEs
	[175]	DRL for computation offloading in IoT devices
	[176]	Stacked autoencoder learning for proactive caching
	[177]	Greedy algorithm to optimize service placement schemes
Base station assoication	[178]	Deep echo state networks for VR application
	[31]	Mean field game with imitation for cell association
Vehicular networks	[179]	Extreme value theory for large queue length prediction
	[180]	Energy demand learning in electric vehicular networks

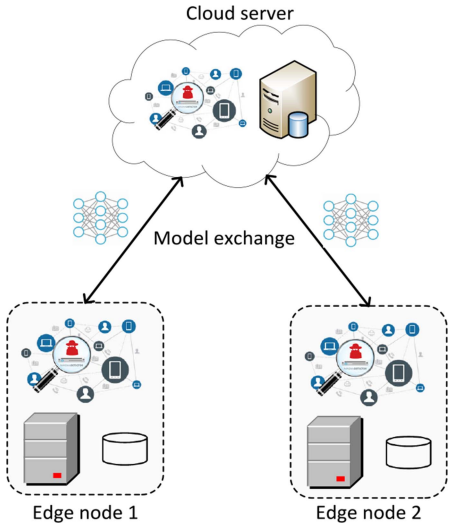


Fig. 14. FL-based attack detection for IoT edge networks.

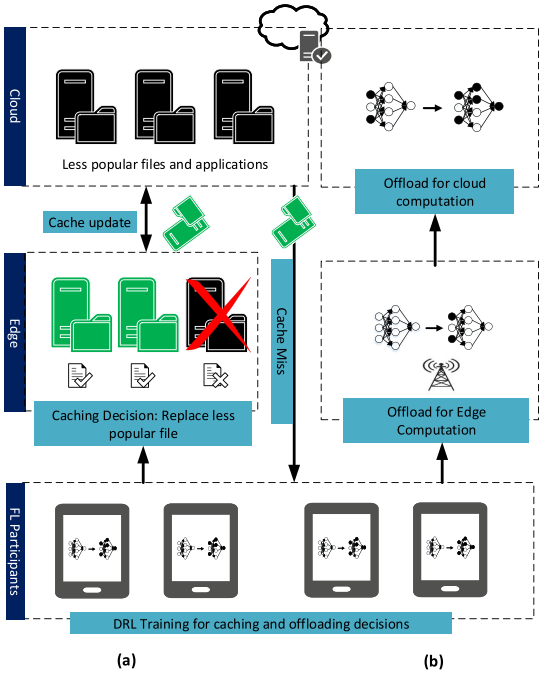


Fig. 15. FL-based (a) caching and (b) computation offloading.

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. 15. 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 [184] can be used to increase the efficiency of training, i.e., training is not initialized from scratch.

Similar to [32], the authors in [175] 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 [185] 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 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 [176] 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 adopted 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 [186]. In addition, privacy of the user is preserved.

The authors in [177] 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 [178] propose an FL based deep echo state networks (ESNs) approach to minimize breaks in presence (BIPs) [187] 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 [188], 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.

Following the ubiquity of IoT devices, the traditional cloud-based approach may no longer be sufficient to cater to dense cellular networks. As computation and storage moves to the edge of networks, the association of users to base stations are increasingly important to facilitate efficient ML model training among the end users. To this end, the authors in [31] 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 [189] 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 occurrence of such low probability events, the authors in [179] propose the use of extreme value theory (EVT) [190]. 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 [191] 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.

Apart from QSI, the vehicles in vehicular networks are also exposed to a wealth of useful captured images that can be adopted to build better inference models, e.g., for traffic optimization. However, these images are sensitive in nature since they can give away the location information of vehicular clients. As such, an FL approach can be used to facilitate collaborative ML while ensuring privacy preservation. However, the images captured by vehicles are often varying in quality due to motion blurs. In addition, another source of heterogeneity is the difference in computing capabilities of vehicles. Given the information asymmetry involved, the authors in [142] propose a multi-dimensional contract design in which the FL server designs contract bundles comprising varying levels of data quality, compute resources, and contractual payoffs. Then, the vehicular client chooses the contract bundle that maximizes its utility, in accordance to its hidden type. Similar to the results in [140], the simulation results show that the FL server derives greatest utility under the proposed contract theoretic approach, in contrast to the linear pricing or Stackelberg game approach.

The authors in [180] 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 [192], 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 [193] based on their physical locations. The clustering-based FEDL reduces the cost of biased prediction [194]. 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 [195]. 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 VIII.

VII. Challenges and Future Research Directions

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

- *Dropped participants:* The approaches discussed in Section IV, e.g., [78], [118], and [119], 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.
- *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 [147], [148], and [149], 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], [154], and [196], and collaborative training, e.g., [155] and [157]. 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.
- *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 [197]. 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. Emerging studies have also considered the use of semi-supervised learning inspired techniques [198].
- *Interference among mobile devices:* The existing resource allocation approaches, e.g., [78] and [119], 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 [127], [129], and [130] 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 edge networks and make optimized decisions.
- *Communication security:* The privacy and security threats studied in Section V-B revolve mainly around data-related compromises, e.g., data and model poisoning. Due to the exposed nature of the wireless medium, FL is also vulnerable to communication security issues such as Distributed Denial-of-Service (DoS) [199] and jamming attacks [200]. 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 [201] such as frequency hopping, e.g., sending one more copy of the model update over different frequencies, can be adopted to address the issue.
- *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 straggler effect. As such, asynchronous FL has been proposed as a solution in [114] and [134]. 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. An approach to be considered is the possibile inclusion of *backup* workers following the studies of [108].
- *Comparisons with other distributed learning methods:* Following the increased scrutiny on data privacy, there has been a growing effort on developing new privacy preserving distributed learning algorithms. One study proposes *split learning* [202], which also enables collaborative ML without requiring the exchange of raw data with an external server. In split learning, each participant first trains the neural network up to a cut layer. Then,

the outputs from training are transmitted to an external server that completes the other layers of training. The resultant gradients are then back propagated up to the cut layer, and eventually returned to the participants to complete the local training. In contrast, FL typically involves the communication of full model parameters. The authors in [203] conduct an empirical comparison between the communication efficiencies of split learning and FL. The simulation results show that split learning performs well when the model size involved is larger, or when there are more participants involved, since the participants do not have to transmit the weights to an aggregating server. However, FL is much easier to implement since the participants and FL server are running the same global model, i.e., the FL server is just in charge of aggregation and thus FL can work with one of the participants serving as the master node. As such, more research efforts can be directed towards guiding system administrators to make an informed decision as to which scenario warrants the use of either learning methods.

- *Further 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.
- *Usage of tools to quantify 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., [204], 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.
- *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 edge server-assisted FL. The combination is able to significantly reduce the size of model updates, as well as the instances of communication with the FL server. However, the feasibility of this combination has not been explored. In addition, the tradeoff 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., data distribution, quantity, number of edge servers, and number of participants.
- *Cooperative mobile crowd ML:* In the existing approaches, mobile devices need 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 [205]. The model exchange between the mobile devices and the cluster head can then 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.
- *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 mainly lies in the federated training of the learning model, with the implementation challenges neglected. For future studies on the applications of FL, besides a need to consider the aforementioned issues in the survey, i.e., communication costs, resource allocation, and privacy and security, there is also a need to consider the specific issues related to the system model in which FL will be adopted in. For example, for delay critical applications, there will be more emphasis on training efficiency and less on energy expense.

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 and future research directions.

References

[1] K. Lueth. *State of the IoT 2018: Number of IoT Devices now at 7B-Market Accelerating*. Accessed: Aug. 19, 2019. [Online]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[2] R. Pryss, M. Reichert, J. Herrmann, B. Langguth, and W. Schlee, "Mobile crowd sensing in clinical and psychological trials—A case study," in *Proc. IEEE Int. Symp. Comput. Based Med. Syst.*, 2015, pp. 23–24.

[3] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 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 crowdsensing," in *Proc. IEEE Sensors Appl. Symp.*, 2015, pp. 1–5.

[6] Y. Jing, B. Guo, Z. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "CrowdTracker: Optimized urban moving object tracking using mobile crowd sensing," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3452–3463, Oct. 2018.

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

[8] W. He, G. Yan, and L. D. Xu, "Developing vehicular data cloud services in the IoT environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1587–1595, May 2014.

[9] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 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*. Hague, The Netherlands: TMC Asser Press, 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 Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[13] G. Ananthanarayanan *et al.*, "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 Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 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," 2019. [Online]. Available: arXiv:1907.08349.

[16] *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper*, Cisco, San Jose, CA, USA, 2016.

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

[18] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

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

[20] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 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. [Online]. Available: arxiv.org/pdf/1602.05629v1

[22] F. Cicirelli *et al.*, "Edge computing and social Internet of Things for large-scale smart environments development," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2557–2571, Aug. 2018.

[23] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: arXiv:1602.05629.

[24] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2018. [Online]. Available: arXiv:1811.03604.

[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," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, Apr. 2018.

[26] K. Powell. (2019). *Nvidia Clara Federated Learning to Deliver AI to Hospitals While Protecting Patient Data*. [Online]. Available: https://blogs.nvidia.com/blog/2019/12/01/clara-federated-learning/

[27] D. Verma, S. Julier, and G. Cirincione, "Federated AI for building AI solutions across multiple agencies," 2018. [Online]. Available: arxiv:1809.10036.

[28] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018.

[29] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[30] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[31] 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 *Proc. IEEE ICC Workshops*, 2018, pp. 1–6.

[32] 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," 2018. [Online]. Available: arXiv:1809.07857.

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

[34] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.

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

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

[37] 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," 2019. [Online]. Available: arXiv:1905.10083.

[38] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 8, pp. 1399–1417, 2018.

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

[40] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[41] 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.

[42] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.

[43] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

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

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

[46] G. Trigeorgis *et al.*, "Adieu features? End-to-end speech emotion recognition using a deep convolutional recurrent network," in *Proc. IEEE ICASSP*, 2016, pp. 5200–5204.

[47] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6A overview of mini-batch gradient descent," *Cited* vol. 14, no. 8, p. 249, 2012.

[48] X. J. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin–Madison, Madison, WI, USA, Rep. TR 1530, 2005.

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

[50] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013. [Online]. Available: arXiv:1312.5602.

[51] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, nos. 4–5, pp. 291–294, 1988.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[53] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

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

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

[56] 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 Process. Mag.*, vol. 35, no. 1, pp. 84–100, Jan. 2018.

[57] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *Int. J. Autom. Comput.*, vol. 14, no. 2, pp. 119–135, 2017.

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

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

[60] 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," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[61] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, 1998.

[62] R. H. Myers and R. H. Myers, *Classical and Modern Regression With Applications*, vol. 2. Belmont, CA, USA: Duxbury Press, 1990.

[63] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, early access.

[64] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," 2018. [Online]. Available: arXiv:1808.07217.

[65] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018. [Online]. Available: arXiv:1806.00582.

[66] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.

[67] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, 2000.

[68] M. Duan, "ASTRAEA: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," 2019. [Online]. Available: arXiv:1907.01132.

[69] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" in *Proc. Int. Conf. Digit. Image Comput. Techn. Appl.*, 2016, pp. 1–6.

[70] J. M. Joyce, "Kullback–Leibler divergence," in *International Encyclopedia of Statistical Science*, Berlin, Germany: Springer, 2011, pp. 720–722.

[71] M. Jaggi *et al.*, "Communication-efficient distributed dual coordinate ascent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3068–3076.

[72] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural Parallel Sci. Comput.*, vol. 11, no. 4, pp. 351–368, 2003.

[73] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019. [Online]. Available: arXiv:1912.00818.

[74] J. Ren, X. Shen, Z. Lin, R. Mech, and D. J. Foran, "Personalized image aesthetics," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 638–647.

[75] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FEDAVG on non-IID data," 2019. [Online]. Available: arXiv:1907.02189.

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

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

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

[79] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1175–1191.

[80] J. Bloemer, "How to share a secret," *Commun. 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," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[82] *TensorFlow Federated: Machine Learning on Decentralized Data*, Google, Mountain View, CA, USA, 2019.

[83] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," 2018. [Online]. Available: arXiv:1811.04017.

[84] S. Caldas *et al.*, "LEAF: A benchmark for federated settings," 2018. [Online]. Available: arXiv:1812.01097.

[85] Y. LeCun, C. Cortes, and C. Burges, *MNIST Handwritten Digit Database*, AT&T Labs, Florham Park, NJ, USA, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

[86] A. Go, R. Bhayani, and L. Huang. (2016). *Sentiment140, Site Functionality, 2013c*. [Online]. Available: http://help.sentiment140.com/site-functionality.Abrufam

[87] WeBank. (2018). *Fate: An Industrial Grade Federated Learning Framework*. [Online]. Available: https://fate.fedai.org

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

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

[90] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[91] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. ICDCS*, 2019, pp. 954–964.

[92] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "ATOMO: Communication-efficient learning via atomic sparsification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9850–9861.

[93] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4447–4458.

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

[95] Z. Tao and Q. Li, "ESGD: Communication efficient distributed deep learning on the edge," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–6.

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

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] Y. Liu *et al.*, "A communication efficient vertical federated learning framework," 2019. [Online]. Available: arXiv:1912.11187.

[99] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, 2018, pp. 1–4.

[100] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-IID data," 2019. [Online]. Available: arXiv:1905.06641.

[101] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 97–105.

[102] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2200–2207.

[103] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: arXiv:1510.00149.

[104] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3329–3337.

[105] 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.

[106] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: An extension of MNIST to handwritten letters," 2017. [Online]. Available: arXiv:1702.05373.

[107] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1488–1492.

[108] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016. [Online]. Available: arXiv:1604.00981.

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

[110] K. Hsieh *et al.*, "GAIA: Geo-distributed machine learning approaching LAN speeds," in *Proc. Symp. Netw. Syst. Design Implement.*, 2017, pp. 629–647.
[111] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. ESANN*, 2013, pp. 1–6.
[112] M. Buscema and S. Terzi, *Semeion Handwritten Digit Data Set*, Center Mach. Learn. Intell. Syst., Univ. California at Irvine, Irvine, CA, USA, 2009.
[113] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," 2019. [Online]. Available: arXiv:1911.02417.
[114] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2018, pp. 21–28.
[115] M. I. Jordan, J. D. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *J. Amer. Stat. Assoc.*, vol. 114, no. 526, pp. 668–681, 2019.
[116] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41–43, 2004.
[117] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," 2019. [Online]. Available: arXiv:1902.00146.
[118] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-FL: Cooperative learning mechanism using non-IID data in wireless networks," 2019. [Online]. Available: arXiv:1905.07210.
[119] 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," 2018. [Online]. Available: arXiv:1812.03633.
[120] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[121] H. T. Nguyen, N. C. Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach," 2019. [Online]. Available: arXiv:1910.09172.
[122] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.
[123] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," 2018. [Online]. Available: arXiv:1808.00023.
[124] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," 2019. [Online]. Available: arXiv:1905.10497.
[125] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," 2020. [Online]. Available: arXiv:2001.07845.
[126] D. López-Pérez, A. Valcarce, G. De La Roche, and J. Zhang, "OFDMA femtocells: A roadmap on interference avoidance," *IEEE Commun. Mag.*, vol. 47, no. 9, pp. 41–48, Sep. 2009.
[127] G. Zhu, Y. Wang, and K. Huang, "Low-latency broadband analog aggregation for federated edge learning," 2018. [Online]. Available: arXiv:1812.11494.
[128] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6463–6486, Oct. 2011.
[129] M. M. Amiri and D. Gunduz, "Federated learning over wireless fading channels," 2019. [Online]. Available: arXiv:1907.09769.
[130] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," 2018. [Online]. Available: arXiv:1812.11750.
[131] 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," 2016. [Online]. Available: arXiv:1609.04836.
[132] P. D. Tao *et al.*, "The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems," *Ann. Oper. Res.*, vol. 133, nos. 1–4, pp. 23–46, 2005.
[133] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, "Approximation bounds for quadratic optimization with homogeneous quadratic constraints," *SIAM J. Optim.*, vol. 18, no. 1, pp. 1–28, 2007.
[134] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019. [Online]. Available: arXiv:1903.03934.
[135] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y.-C. Liang, "Joint service pricing and cooperative relay communication for federated learning," 2018. [Online]. Available: arXiv:1811.12082.
[136] M. J. Osborne *et al.*, *An Introduction to Game Theory*, vol. 3. New York, NY, USA: Oxford Univ. Press, 2004.
[137] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A Stackelberg game perspective," 2019. [Online]. Available: arXiv:1908.03092.
[138] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, early access, Jan. 20, 2020, doi: 10.1109/JIOT.2020.2967772.
[139] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," 2019. [Online]. Available: arXiv:1911.05642.
[140] 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," 2019. [Online]. Available: arXiv:1905.07479.
[141] P. Bolton and M. Dewatripont, *Contract Theory*. Cambridge, MA, USA: MIT Press, 2005.
[142] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.
[143] R. Dennis and G. Owen, "Rep on the block: A next generation reputation system based on the blockchain," in *Proc. Int. Conf. Internet Technol. Secured Trans.*, 2015, pp. 131–138.
[144] H. Yu *et al.*, "A fairness-aware incentive scheme for federated learning," in *Proc. AAAI/ACM Conf. AI Ethics Soc.*, 2020, pp. 393–399.
[145] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 691–706.
[146] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *Proc. IEEE ICIP*, 2014, pp. 343–347.
[147] 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," *Int. J. Security*, vol. 10, no. 3, pp. 137–150, 2015.
[148] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.
[149] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIS," in *Proc. Security Symp.*, 2016, pp. 601–618.
[150] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 3–18.
[151] N. Papernot, P. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016. [Online]. Available: arXiv:1605.07277.
[152] P. Laskov *et al.*, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Security Privacy*, 2014, pp. 197–211.
[153] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptography Conf.*, 2006, pp. 265–284.
[154] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017. [Online]. Available: arXiv:1712.07557.
[155] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.
[156] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[157] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.
[158] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[159] Y. Liu, Z. Ma, S. Ma, S. Nepal, and R. Deng. (2019). *Boosting Privately: Privacy-Preserving Federated Extreme Boosting for Mobile Crowdsensing*. [Online]. Available: https://arxiv.org/pdf/1907.10218.pdf
[160] A. Triastcyn and B. Faltings, "Federated generative privacy," 2019. [Online]. Available: arXiv:1910.08385.
[161] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[162] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proc. IEEE ICC*, 2019, pp. 1–6.
[163] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017. [Online]. Available: arXiv:1712.05526.
[164] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018. [Online]. Available: arXiv:1808.04866.
[165] D. Dua and C. Graff, *UCI Machine Learning Repository*, School Inf. Comput. Sci., Univ. California Irvine, Irvine, CA, USA, 2019. [Online]. Available: http://archive.ics.uci.edu/ml
[166] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," 2018. [Online]. Available: arXiv:1811.12470.
[167] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2018. [Online]. Available: arXiv:1807.00459.
[168] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," 2018. [Online]. Available: arXiv:1808.03949.
[169] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Depend. Secure Comput.*, early access, Nov. 8, 2019, doi: 10.1109/TDSC.2019.2952332.
[170] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput. Pract. Exp.*, vol. 28, no. 10, pp. 2991–3005, 2016.
[171] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of Vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, 2014, pp. 241–246.
[172] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in fog-to-things computing," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 169–175, Feb. 2018.
[173] T. D. Nguyen *et al.*, "D_IoT: A crowdsourced self-learning approach for detecting compromised IoT devices," 2018. [Online]. Available: arXiv:1804.07474.
[174] 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," *Appl. Sci.*, vol. 8, no. 12, p. 2663, 2018.
[175] 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. 69194–69201, 2019.
[176] Z. Yu *et al.*, "Federated learning based proactive content caching in edge computing," in *Proc. IEEE GLOBECOM*, 2018, pp. 1–6.
[177] 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," *Inf. Sci.*, vol. 505, pp. 562–570, Dec. 2019.
[178] M. Chen, O. Semiari, W. Saad, X. Liu, and C. Yin, "Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks," 2018. [Online]. Available: arXiv:1812.01202.
[179] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," in *Proc. IEEE GLOBECOM*, 2018, pp. 1–7.
[180] Y. Saputra, H. Dinh, D. Nguyen, E. Dutkiewicz, M. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *Proc. IEEE GLOBECOM*, 2019, pp. 1–6.
[181] 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 *Proc. IEEE WCNC*, 2018, pp. 1–6.
[182] TU New Brunswick. *NSL-KDD*. Accessed: Aug. 25, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html
[183] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–6.
[184] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[185] S. Priya and D. J. Inman, *Energy Harvesting Technologies*, vol. 21. New York, NY, USA: Springer, 2009, p. 2.
[186] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2249–2257.
[187] J. Chung, H.-J. Yoon, and H. J. Gardner, "Analysis of break in presence during game play using a linear mixed model," *ETRI J.*, vol. 32, no. 5, pp. 687–694, 2010.
[188] 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 J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.
[189] O. Guéant, J.-M. Lasry, and P.-L. Lions, "Mean field games and applications," in *Paris–Princeton Lectures on Mathematical Finance 2010*. Heidelberg, Germany: Springer, 2011, pp. 205–266.
[190] L. De Haan and A. Ferreira, *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 2007.
[191] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[192] P. You and Z. Yang, "Efficient optimal scheduling of charging station with multiple electric vehicles via V2V," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, 2014, pp. 716–721.
[193] P. Bradley, K. Bennett, and A. Demiriz, *Constrained K-Means Clustering*, vol. 20, Microsoft Res., Redmond, WA, USA, 2000.
[194] W. Li, T. Logenthiran, V.-T. Phan, and W. L. Woo, "Implemented IoT-based self-learning home management system (SHMS) for Singapore," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2212–2219, Jun. 2018.
[195] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, p. 16, 2018.
[196] L. Jiang, R. Tan, X. Lou, and G. Lin, "On lightweight privacy-preserving collaborative learning for Internet-of-Things objects," in *Proc. IoTDI*, 2019, pp. 70–81.
[197] Z. Gu *et al.*, "Reaching data confidentiality and model accountability on the Caltrain," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2019, pp. 336–348.
[198] A. Albaseer, B. S. Ciftler, M. Abdallah, and A. Al-Fuqaha, "Exploiting unlabeled data in smart cities using federated learning," 2020. [Online]. Available: arXiv:2001.04030.
[199] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 3. Nashville, TN, USA, Oct. 2000, pp. 2275–2280.
[200] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: Attack and defense strategies," *IEEE Netw.*, vol. 20, no. 3, pp. 41–47, May/Jun. 2006.
[201] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *Proc. IEEE Symp. Security Privacy*, 2008, pp. 64–78.
[202] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018. [Online]. Available: arXiv:1812.00564.
[203] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," 2019. [Online]. Available: arXiv:1909.09145.
[204] I. I. Eliazar and I. M. Sokolov, "Measuring statistical heterogeneity: The Pietra index," *Physica A Stat. Mech. Appl.*, vol. 389, no. 1, pp. 117–125, 2010.
[205] 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 Commun. Lett.*, vol. 19, no. 2, pp. 259–262, Feb. 2015.

Wei Yang Bryan Lim received the undergraduate degree (Double First Class Hons.) in economics and business administration (finance) from the National University of Singapore in 2018. He is currently pursuing the Ph.D. degree with the Alibaba Group and Alibaba-NTU Joint Research Institute, Nanyang Technological University, Singapore. His research interests include federated learning and edge intelligence.

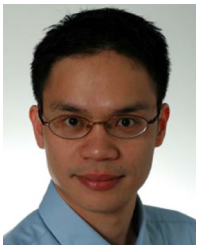
Nguyen Cong Luong is currently a Lecturer with the Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam. He is also a Researcher with the Phenikaa Research and Technology Institute (PRATI), Hanoi. His research interests include signal processing and resource management in wireless networks.

Dinh Thai Hoang (Member, IEEE) is currently a Faculty Member with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. He received the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2016. His research interests include emerging topics in wireless communications and networking, such as ambient backscatter communications, vehicular communications, cybersecurity, IoT, and 5G networks. He is currently an Editor of IEEE WIRELESS COMMUNICATIONS LETTERS and the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

Yutao Jiao is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests are mainly in resource allocation and mechanism design in various topics, including big data analysis, machine learning, and blockchain.

Ying-Chang Liang (Fellow, IEEE) was a Professor with the University of Sydney, Australia, a Principal Scientist and a Technical Advisor with the Institute for Infocomm Research, Singapore, and a Visiting Scholar with Stanford University, USA. He is currently a Professor with the University of Electronic Science and Technology of China, China, where he leads the Center for Intelligent Networking and Communications. He serves as the Deputy Director of the Artificial Intelligence Research Institute. His research interests include wireless networking and communications, cognitive radio, symbiotic networks, dynamic spectrum access, the Internet of Things, artificial intelligence, and machine learning techniques. He is a Foreign Member of Academia Europaea. He has been recognized by Thomson Reuters (currently, Clarivate Analytics) as a Highly Cited Researcher, since 2014. He received the Prestigious Engineering Achievement Award from the Institution of Engineers, Singapore, in 2007, the Outstanding Contribution Appreciation Award from the IEEE Standards Association in 2011, and the Recognition Award from the IEEE Communications Society Technical Committee on Cognitive Networks in 2018. He was a recipient of numerous paper awards, including the IEEE Jack Neubauer Memorial Award in 2014, and the IEEE Communications Society APB Outstanding Paper Award in 2012. He was the Chair of the IEEE Communications Society Technical Committee on Cognitive Networks, and served as a TPC Chair and an Executive Co-Chair for the IEEE Globecom'17. He was also an Associate Editor-in-Chief of *Random Matrices: Theory and Applications* (World Scientific). He is the Founding Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (Cognitive Radio Series), and the Key Founder and the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He is also serving as the Associate Editor-in-Chief for *China Communications*. He served as the Guest/Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS, the *IEEE Signal Processing Magazine*, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. He was a Distinguished Lecturer of the IEEE Communications Society and the IEEE Vehicular Technology Society.

Qiang Yang (Fellow, IEEE) received the B.Sc. degree in astrophysics from Peking University in 1982, the M.Sc. degree in astrophysics from the University of Maryland, College Park, in 1985, and the Ph.D. degree from the Computer Science Department, University of Maryland in 1989. He was a Faculty Member with the University of Waterloo from 1989 to 1995 and Simon Fraser University from 1995 to 2001. He is the Head of AI, WeBank (Chief AI Officer) and a Chair Professor with the Computer Science and Engineering (CSE) Department, Hong Kong University of Science and Technology, where he was a Former Head of CSE Department and the Founding Director of the Big Data Institute from 2015 to 2018. He has authored several books, including *Transfer Learning* (Cambridge Press), *Federated Learning* (Morgan Claypool), *Intelligent Planning* (Springer), *Crafting Your Research Future* (Morgan Claypool), and *Constraint-Based Design Recovery for Software Engineering* (Springer). His research interests include artificial intelligence, machine learning and data mining, especially in transfer learning, automated planning, federated learning, and case-based reasoning. He was the Founding Editor in Chief of the *ACM Transactions on Intelligent Systems and Technology* and the IEEE TRANSACTIONS ON BIG DATA. He served as the President of International Joint Conference on AI (IJCAI, from 2017 to 2019) and an Executive Council Member of the Association for the Advancement of AI (AAAI, from 2016 to 2020). He is a recipient of several awards, including the 2004/2005 ACM KDDCUP Championship, the ACM SIGKDD Distinguished Service Award in 2017, and the AAAI Innovative AI Applications Award in 2016 and 2019. He was the Founding Director of Huawei's Noah's Ark Lab from 2012 to 2014 and a Co-Founder of 4Paradigm Corporation, an AI platform company. He is a fellow of several international societies, including ACM, AAAI, IAPR, and AAAS.



Dusit Niyato (Fellow, IEEE) received the B.E. degree from the King Mongkuk's Institute of Technology Ladkrabang, Thailand, in 1999, and the Ph.D. degree in electrical and computer engineering from the University of Manitoba, Canada, in 2008. He is currently a Professor with the School of Computer Science and Engineering and the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. He has published more than 380 technical papers in the area of wireless and mobile networking, and is an inventor of four U.S. and German patents. He has authored four books, including "Game Theory in Wireless and Communication Networks: Theory, Models, and Applications" (Cambridge University Press). He won the Best Young Researcher Award of IEEE Communications Society Asia Pacific and the 2011 IEEE Communications Society Fred W. Ellersick Prize Paper Award. He was named the 2017, 2018, 2019 highly cited researcher in computer science. He is currently serving as a Senior Editor of IEEE WIRELESS COMMUNICATIONS LETTER, an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (Radio Management and Multiple Access) and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (Network and Service Management and Green Communication), an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS, an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He was a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS ON COMMUNICATIONS. He was a Distinguished Lecturer of the IEEE Communications Society from 2016 to 2017.

Chunyan Miao (Senior Member, IEEE) received the B.S. degree from Shandong University, Jinan, China, in 1988, and the M.S. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1998 and 2003, respectively, where she is currently a Professor with the School of Computer Science and Engineering and the Director of the Joint NTU–UBC Research Centre of Excellence in Active Living for the Elderly (LILY). Her research focus on infusing intelligent agents into interactive new media (virtual, mixed, mobile, and pervasive media) to create novel experiences and dimensions in game design, interactive narrative, and other real world agent systems.