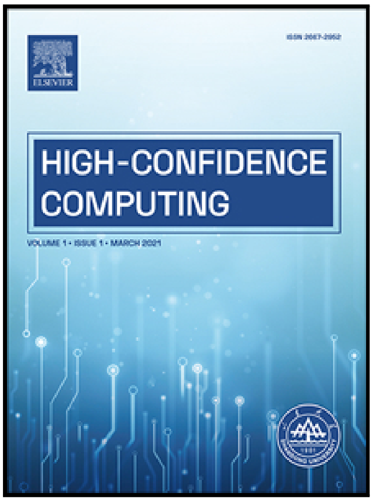

A Survey of Federated Learning for Edge Computing: Research Problems and Solutions

Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, Qun Li

PII: S2667-2952(21)00009-X
DOI: https://doi.org/10.1016/j.hcc.2021.100008
Reference: HCC 100008

To appear in: *High-Confidence Computing*

Received date: 24 December 2020
Revised date: 3 March 2021
Accepted date: 3 March 2021

Please cite this article as: Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, Qun Li, A Survey of Federated Learning for Edge Computing: Research Problems and Solutions, *High-Confidence Computing* (2021), doi: https://doi.org/10.1016/j.hcc.2021.100008

# A Survey of Federated Learning for Edge Computing: Research Problems and Solutions[☆]

Qi Xia*, Winson Ye, Zeyi Tao, Jindi Wu, Qun Li

*Department of Computer Science, College of William and Mary*
*251 Jamestown Rd., Williamsburg, VA 23185, USA*

**Abstract**

Federated Learning is a machine learning scheme in which a shared prediction model can be collaboratively learned by a number of distributed nodes using their locally stored data. It can provide better data privacy because training data are not transmitted to a central server. Federated learning is well suited for edge computing applications and can leverage the the computation power of edge servers and the data collected on widely dispersed edge devices. To build such an edge federated learning system, we need to tackle a number of technical challenges. In this survey, we provide a new perspective on the applications, development tools, communication efficiency, security & privacy, migration and scheduling in edge federated learning.

*Keywords:* Federated learning, Edge computing

## 1. Introduction

The proliferation of real time technologies such as VR, AR, and self-driving cars has led researchers and industry executives to come up with new architecture for data processing. The traditional model of cloud computing is unsuitable for applications that demand low latency, so as a result a new model of computation termed edge computing has sprung forth. Edge computing is primarily concerned with transmitting data among the devices at the edge, closer to where user applications are located, rather than to a centralized server (see Figure 1). Edge node (or edge client, edge device) is usually the resource-constraint device that end user uses and it is geographically close to the nearest edge server, who has abundant computing resources and high bandwidth communicating with

[☆] This project was supported in part by US National Science Foundation grant CNS-1816399. This work was also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

[*] Corresponding author

*Email addresses:* qxia@cs.wm.edu (Qi Xia), wye@email.wm.edu (Winson Ye), ztao@cs.wm.edu (Zeyi Tao), jwu21@email.wm.edu (Jindi Wu), liqun@cs.wm.edu (Qun Li) end nodes. When the edge server requires more computing power, it will connect to the cloud server. The most important consequences of this architecture are twofold: latency is dramatically reduced as data does not need to travel as far, and bandwidth availability improves significantly, as the user is no longer relying on sharing a single traffic lane in order to transfer their data. Indeed, this new computing paradigm offers great cost savings for companies who do not have the resources to build dedicated data centers for their operations. Instead, engineers can build a reliable network of smaller and cheaper edge devices.

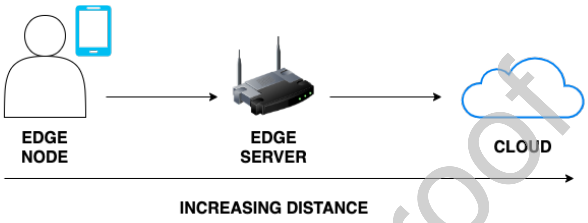

Figure 1: Devices by distance to user.

In addition, federated learning has been discussed a lot recently. It is a collaborative machine learning framework allowing devices from different resources with different private datasets working together to study and train a global model. Federated learning can not only collaborate the computational resources from different devices, but also preserve the privacy at the same time.

Given the common features of both edge computing and federated learning, edge computing is a naturally suitable environment to apply federated learning framework. Therefore, edge federated learning is more and more appealing in both academic research and industry in recent days. Here, we first have a brief introduction to edge computing and federated learning respectively and discuss about their key advantages.

### 1.1. Edge Computing

There are a set of key reasons why industry executives are transitioning from a traditional cloud-based model to edge computing platforms. The two major factors that were already discussed beforehand are low latency and high bandwidth [1]. However, the edge also provides for greater security. For example, sending data to an edge device will give any potential attackers less time to launch an attack as compared to the cloud simply because the latency is lower. Moreover, attacks like DDoS that would normally be debilitating in a cloud-based environment are rendered almost harmless in an edge computing environment because the affected edge devices can be removed from the network without hampering the overall functionality of the network as a whole. Of course, this also means that edge networks are much more reliable as they do not have a single point of failure. As discussed briefly beforehand, edge networks are much more easily scalable because the devices have much smaller footprints. Indeed, a scale-out strategy of scalability rather than a scale-up one offers companies a very attractive way of getting good performance with low cost. Moreover, some of these edge devices or edge data centers may not even need to be built from scratch by any one company. Different stakeholders can partner up to share the resources from the already existing IoT devices in the edge network.

In order to deliver these benefits to end-users, engineers have relied on a common set of key operating principles when building edge computing systems [2]:

- Mobility: For applications like self-driving cars, the edge devices have to accommodate a constantly moving end-user without sacrificing latency or bandwidth. Some approaches solve this problem by positioning edge devices on the roadside.
- Proximity: In order to deliver low latency guarantees, the edge devices must be positioned as close as possible to the end users. This could mean performing computation directly at the edge device or investing in a local edge computing data center that is close to the end-user.
- Coverage: For edge computing to become ubiquitous, network coverage must be far-reaching. Thus, the exact distribution of nodes in an edge computing framework is imperative to achieving an optimal user experience. Of course, a dense distribution is preferred, but this must be balanced with cost constraints.

### *1.2. Federated Learning*

Federated learning is a method for training neural networks across many devices. In this model of computation, a single global neural network is stored in a central server. The data used to train the neural network is stored locally across multiple nodes and are usually heterogeneous. Here we assume we have several nodes $node_1$, $node_2, \cdots, node_n$. On the node side, $node_i$ keeps the private dataset $\xi_i$. If we assume the loss function on the neural network is $f(\cdot)$, in one synchronization, $node_i$ computes the updated weight based on the current weight at time $t$ $w_t^i$, step size at time $t$ as $\gamma_t$, and its private dataset $\xi_i$:

$$w_{t+1}^i = w_t - \gamma_t \cdot \frac{\partial f(w_t, \xi_i)}{\partial w} \quad (i = 1, 2, \cdots, n) \tag{1}$$

Note that this local update can run one or several iterations. On the server side, it receives the weights that are uploaded by all the nodes. Central server uses an aggregation function $A(\cdot)$ to aggregate all the uploaded weights and update the weights for the next round. The updated weights at time $t + 1$ are:

$$w_{t+1} = A(w_t^1, w_t^2, \cdots, w_t^n) \tag{2}$$

In practice, we usually simply use an average function to aggregate the uploaded weights and update the global model. The model is replicated across all the end devices as needed so predictions can be made locally. Because of the heterogeneity of federated learning, we do not require all $n$ nodes to participate in one synchronization. Only some of the nodes will be randomly selected to perform the computation.

Note that federated learning is distinct from the traditional distributed computing scenario. The most profound difference lies in the assumptions made on the datasets. In distributed learning, the partitions of the dataset are assumed to be i.i.d., meaning that they are generated from the same memoryless stochastic process. However, no such assumption is made in the federated learning setting [3]. Instead, datasets can be heterogeneous. For example, an ML model designed to recognize criminals within a neighborhood may rely on camera footage collected by a diverse group of users. Clearly, one cannot reasonably expect the footage collected between two users to be i.i.d.

The promise of federated learning is appealing to many users. There are a number of key advantages to take note of:

- Training time is reduced. Multiple devices are used to calculate gradients in parallel, which offers significant speedups.
- Inference time is reduced. At the end of the day, each device has their own local copy of the model, so predictions can be made extremely quickly and without having to rely on slow queries to the cloud.
- Privacy is preserved. Uploading sensitive information to the cloud presents a major privacy risk for applications like healthcare devices. Privacy breaches in these settings may literally be a matter of life and death. As such, keeping data local helps preserve the privacy of end users.
- Collaborative learning is easier. Instead of having to collect one massive dataset to train a machine learning model, federated learning allows for a ”crowdsourcing” of sorts that can make the data collection and labeling process much easier in terms of time and effort spent.

Because of the natural advantages of both edge computing and federated learning, and the fact that edge computing is a very suitable environment to deploy federated learning, the promising future of edge federated learning prompts us to present a survey designed to explore the research problems that arise in this new area. The rest of this paper will detail these challenges, summarize how the state of the art solves them, as well as provide our own insights into the future of this field. In this article, first, we will go into details regarding how different applications make use of edge federated learning frameworks. Second, we will discuss existing programming models for edge federated learning. Third, we will discuss communication and computation efficiency. Fourth, we will discuss security and privacy. In the end, we will discuss resource allocation and migration.

## 2. Applications

Edge federated learning solves the data island problem by fully exploring the huge potential of the data on terminal devices without infringing on user's privacy, and it greatly improves the efficiency of model learning in edge computing systems. Therefore, it can be widely used in many scenarios where privacy protection and resource utilization are critical. In this section, we will discuss a few scenarios for edge federated learning, and some recent work applied in these scenarios.

### *2.1. Healthcare System*

The excellent performance of deep learning in complex pattern recognition tasks makes it more widely used in the medical industry. For each medical institution, its data is separately stored and processed in the edge node, but the model trained with a small dataset that is collected from an individual medical institution does not have a satisfactory accuracy when it is applied to the unseen data that is somehow uncorrelated with the training data. Therefore, a large amount of real electronic health records (EHR) is needed to train a powerful medical model. However, the demand for real dataset is hard to satisfy because of the sensitivity and privacy of medical data. Edge federated learning can help overcome this problem, allowing medical institutions to collaborate on training models without sharing patient data so that they can meet the requirements of data privacy protection and the Health Insurance Portability and Accountability Act (HIPAA). For example, Liu et al. trained a chest X-ray image classification model using federated learning for COVID-19 [4]. Sheller et al. used edge federated learning to train an image semantic segmentation model for brain scans and the simulation result shows that the performance of the proposed model is similar to the model trained with shared data [5]. Their extended work applied the final model selection mechanism in which each medical institution selects the best locally validated model for global model aggregation to achieve better performance for the medical image learning model [6].

In addition to medical institutions, the edge federated transfer learning method is applied to personal health measurement devices. Some personal healthcare devices, such as blood pressure meter and activity recognition device, are utilized to observe health conditions and push health alarms in time, which plays an important role in smart health systems [7]. For users, it is necessary to have a ready-made model at the beginning and train a personalized model updated by their physical conditions in real time. Chen et al. proposed an accurate and personalized healthcare model FedHealth [8]. Moreover, FedPer [9] and pFedMe [6] can be used to collaboratively learn a model at the network edge while capturing personalization.

### *2.2. Vehicular Network*

The data generated by the device on vehicles such as location and orientation detected by the GPS, images captured by the on-board camera, and the pressure data from the oil pressure sensor, are valuable resources for vehicle manufacturers to provide intelligent navigation services and early warnings. The on-board computer collects locally generated sensing data, then uploads it to the Vehicle Edge Computing (VEC) system to train the local learning model. Edge federated learning in VEC can meet the needs of users for smart vehicle decision-making. For instance, a clustering-based federated energy demand learning approach is implemented by Saputra et al. in [10] for electric vehicle networks to make energy demand prediction in the considered areas.

In addition, image classification is a typical task in vehicular networks. The performance and efficiency of edge federated learning are highly impacted by the training data quality and the computational power of edge nodes, respectively. Ye et al. proposed a selective model aggregation approach [11], in which a model is selected if its training images are in high quality and the edge node has sufficient computation capability. In order to further improve the learning accuracy and encourage devices with high-quality data to join the model training process, Kang et al. design an incentive mechanism [12] using the contract theory.

Moreover, autonomous vehicles are equipped with more sensors than regular vehicles such as LiDAR and ultrasonic sensors to perceive the surrounding environment without human interaction. Edge federated learning is a desirable solution in the VEC system to learn a privacy-preserving machine learning model from non-IID vehicular data [13].

### *2.3. Intelligent Recommendation*

Intelligent recommendation is a useful function in smartphone or desktop applications to predict user choices so that users can easily access and use it. Compared with standard machine learning approaches, edge federated learning is capable of effectively training flexible models for recommendation tasks. Because edge nodes are located in a certain area and have similar tasks for efficiency and cost reasons, this kind of similarity among edge nodes can be used to train adaptive models by edge federated learning. For instance, the researchers from Google Keyboard (Gboard) team train models using edge federated learning on a global scale for virtual keyboard search suggestion [14] and emoji prediction [15], and the evaluation results show that the models on each edge node have a good performance because the models are adjusted to different language and culture styles in a specific area. In addition, Hartmann et al. show that the browser option suggestion model trained with federated learning can help users quickly find the website they need by entering fewer characters [16]. The work can be improved in edge federated learning systems to provide different users with relatively personalized models by exploring user similarities without violating user privacy.

## 3. Development Tools

There are many concerns that the programmer needs to take into account when designing an edge federated learning system. Issues such as different

APIs, dataflow models, network configurations, and device properties have to be considered. In light of the complexity involved in edge federated learning, it is important that the research community spend time developing tools that can help programmers build edge federated learning systems more easily. In this section, we will be discussing the following areas that could benefit from development tools:

- Application-level support. This is concerned with providing easy to use APIs for the developer.
- Systems design Support. This is concerned with providing helpful abstractions for systems level technicalities such as network configuration.

### *3.1. Application-Level Support*

First, let us discuss the available application-level support for edge federated learning systems. Ideally, any application-level support would come in the form of easy-to-use integrated development environments or APIs that can help the average developer perform common edge federated learning tasks easily. The reader can think back to the many classical software engineering tools such as numpy for numerical processing or IDLE for Python programming as examples of good application-level support tools.

One work of note is the programming model proposed by Hong et al [17]. The authors provide a set of event handlers that the programmer must implement and functions that individual applications can call upon. This way, whenever a significant event occurs, such as when a message arrives from another device, the programmer can rest assured that the event handlers will do most of the work. For federated learning in particular, some adjustments may need to be made. For example, federated learning systems usually require aggregation functions in order to assemble all the local gradients. This would have to be provided in the API. Event handlers would also have to be implemented to facilitate different stages of the learning process, such as when a round of learning has finished. However, given the easily extensible nature of their framework, we believe that it would be fairly straightforward to implement these changes for federated learning systems.

Another significant paper by Giang et al. [18] focuses on developing a good abstraction that allows developers to reason about the complexities of edge federated learning more easily. In particular, they propose a methodology for federated learning systems using dataflow graphs. Even though this idea was proposed for edge computing, it is possible to generalize this kind of framework to edge federated learning systems. Their dataflow program can handle three key issues: 1) heterogeneity, 2) mobility, and 3) scalability. For example, to avoid vertical and horizontal heterogeneity, the program contains specialized nodes developed by domain experts that can only be wired together with specific nodes. Mobility requirements can be fulfilled through code duplication. Scalability requirements are met by eliminating the need for an internal management system to coordinate communication between nodes.

### *3.2. Systems Design Support*

Next, let us discuss the available development tools for system level design. Ideally, we are looking for tools that can help developers accomplish systems-level tasks such as load balancing, resource management, or migrations easily. Some of these features may be integrated into a larger IDE designed for edge federated learning. For example, a tool akin to MapReduce would be very helpful in the edge federated learning setting.

The primary work here is Bonawitz et al.'s work based on TensorFlow [19]. The major contribution of their work is offering a mature systems level framework that the developer can use to deploy their federated learning applications. They deal with a variety of key issues: 1) device availability, 2) resource management, and 3) reliability. In a federated learning setting, devices cannot be expected to always be available for a given round. The authors implement a "pace steering" mechanism that allows the server to suggest reconnection times that will ensure a sufficient number of devices are connected to ensure progress in the learning task. In order to handle the limited resources issues in end devices, the authors created their framework such that a learning job is only run on the cellphone when it is idle, charging, and connected to WiFi. Besides, to deal with the limited storage capabilities of cellphones, the authors provide programming utilities to help minimize the storage footprint of the local data these devices must store in order to participate in the federated learning task. As for the reliability, the authors maintain a coordinator to to deal with issues such as data crash, learning fail, etc.

### *3.3. Future Directions*

In the future, we believe that the research community should focus their attention on three key areas: 1) containerization, 2) security frameworks, and 3) extending current edge computing programming models to the edge federated learning setting.

By containerization, we are referring to the management of all the various execution environments that devices in the edge federated learning setting will utilize. Since many of the devices in edge federated learning may be IoT devices, it is important that operating systems remain as lightweight as possible. Several key challenges are present in this area. For example, how will programmers manage all the containers running on heterogeneous devices? How will these containers communicate with the outside world without exposing themselves to any security vulnerabilities? Encouraging progress has already been made in the industry. For example, WeBank's KubeFate[1][2] allows developers to run federated learning tasks across multiple containers, with features like security and privacy already built into the framework. Other notable examples include TensorFlow

[1] FATE: Federated AI Technology Enabler, https://github.com/FederatedAI/FATE

[2] KubeFATE: https://github.com/FederatedAI/KubeFATE

Federated[3], PySyft[4], and PaddleFL[5]. While all these applications do address the containerization issue, they are not yet mature technologies. For example, many of these applications reply on Kubernetes for container orchestration, which some users may take issue with because of the high overhead.

Second, researchers should consider building security APIs that programmers can utilize in order to secure their own edge federated learning systems. Current theory on the subject covers ideas such as differential privacy, homomorphic encryption, multi-party computation, and secure enclaves. Nevertheless, major challenges exist when it comes to implementing these security measures. For example, Kairouz et al. note that there is not yet a methodology for distributing federated learning functions across trusted execution environments [20].

Finally, as the reader may have noticed in the previous sections, many of the previous work referenced do not pertain directly to edge federated learning. Instead, they refer to edge computing systems in general. As such, it is imperative that researchers focus on extending these tools to the edge federated learning setting. Edge federated learning is unique in that machine learning tasks require massive computations as well as storage capabilities. For the model to perform well, gradients must be coordinated carefully as the sudden failure of a few devices or the presence of malicious actors may cause the model to act unexpectedly.

## 4. Communication and Computation Efficient Edge Federated Learning

Edge federated learning is a privacy-preserving machine learning framework where the data is distributed across many resource constrained edge devices. It shares the same training procedure as baseline federated learning [21] that is an edge server that distributes an initial model to each edge node who independently updates the model (local model) via local data, and the global model is updated by aggregating a subset of the local models. The server broadcasts this new global model to all nodes to start a new round of local training. This training procedure repeats until some criterion meets.

### *4.1. Scale of Federation*

Similar to the conventional federated learning [22], the edge federated learning can also be categorized into two types by the scale of federation: cross-silo and cross-device edge federated learning. Cross-silo edge federated learning trains data from different organizations (e.g. medical center or geo-distributed datacenter). On the other hand, cross-device federated learning trains data on

[3] TensorFlow Federated: https://github.com/tensorflow/federated

[4] PySyft: https://github.com/OpenMined/PySyft

[5] PaddleFL: https://github.com/PaddlePaddle/PaddleFL

many IoT devices. The major difference between them is the number of participating training nodes and the amount of training data stored on each node. In this section, we discuss the impact of the scale of federation and how it affects communication and computation cost on edge federated learning.

### *4.1.1. Cross-device Edge Federated Learning*

In cross-device edge federated learning, the number of active training nodes is in the order of millions and each node has relatively small amounts of data as well as computational power [23]. The nodes are usually portable devices or sensors. A remarkable example here is improving the query suggestion of Google Keyboard by [14]. The major challenges that cross-device edge federated learning faces are:

- There are extremely high communication costs when edge servers synchronize the training models and broadcast a new global model to each node for next step training.
- It is hard to efficiently manage a large number of nodes and deal with possible issues such as the unexpected network connectivity between node and server.

Given the number of total nodes $\mathcal{N}$, selection rate $\eta$, the total communication cost for one round training can be formulated as

$$2 \cdot \tau \cdot \mathcal{N} \cdot \eta \cdot \mathcal{M} \tag{3}$$

where $\tau$ indicates the number of global synchronizations that model can converge and $\mathcal{M}$ is the raw size of the training model including all weights and training metadata. For the sake of simplicity, we denote $\mathcal{M}$ as the number of total trainable parameters $P_n$ multiplies its precision such as $\mathcal{M} = P_n \cdot \text{bit}$ (4, 8, 16, 32). For example, as the winner of ILSVRC-2012 competition, AlexNet [24] comes along with nearly 61 million 32-bit real value parameters with an actual model size of 233MB. In the original federated learning, model aggregation happens on every global synchronization and it requires selected nodes passing their local models $W_t^k$ where $k \in [N]$ to the central server. This setting can be further relaxed to be only passing the local updates $\Delta W_t^k = W_{t+e}^k - W_t^k$ where $e$ is the number of local epochs. The keys to reduce the communication cost in edge federated learning as shown in Equation 3 are total communication rounds $\tau$ and model size $\mathcal{M}$.

To reduce the communication cost in edge federated learning, one can reduce the size of local update $\Delta W_t^k$ by either vector quantization or specification. On the other hand, we can also find the optimal choice of $\tau$ for minimizing the overall communication cost of the process. The former has been widely studied in the past decade. However, determining the optimal communication rounds $\tau$ seems tricky. This is due to:

- Increasing the number of training nodes can significantly put the negative influence on performance of the model therefore it requires more global synchronization rounds to meet certain criterion.

- Training on highly decentralized and heterogeneous data makes the contribution of each local model shard to the global mode to be rather limited which prolong the training epochs.

Although increasing the local training epochs may mitigate this issue, it introduces extra computational workload and power consumption on each node. Finding optimal communication iterations remains an open problem.

### *4.1.2. Cross-silo Edge Federated Learning*

In cross-silo edge federated learning, on the contrary, the number of nodes is relatively small, but it requires the nodes to have sufficient computational resources for processing a huge amount of data on each edge server. For example, big online retailers would recommend items for users by training tens of million shopping data stored in geo-distributed data centers. In the above settings, the challenge is how edge federated learning efficiently distributes computation to edge servers under the constraint of computation budgets and privacy models. In recent years, many researchers tend to deploy large and powerful deep networks to resource constrained devices because deeper and wider networks usually achieve better performance than shallow networks [25]. It also relies on powerful end devices.

Given a certain level computation budget, network quantization and pruning can further expand the reach of edge federated learning. Quantizing network weights with a small number of bits can significantly accelerate network training and inference as well as reduce model size. In [26], researchers reduce float version VGG-19 [27] from $\sim 500M$ to $\sim 32M$ with ternary precision and no accuracy loss. Many network quantization mainly focuses on optimization at a single model. The quantized weights are only used during the forward and backward propagations but not during the parameter updates. It is very obvious that passing the non-quantized updates in edge federated learning is undesired. A good starting point is from [28], authors introduce an algorithm allowing model updates to be quantized before being transmitted but they do not use quantized training models. One open question here is that can we train quantized models in edge federated learning while using quantized model updates at the same time such that we can achieve both communication and computation efficient edge federated learning training. Network pruning can be very beneficial, and it can also efficiently reduce the complexity of neural network models. A common approach in network pruning is dropping the parameter with small enough magnitude. Similar to network quantization, the existing network pruning algorithms are also limited to the single non-distributed settings, and neural networks are usually pruned step by step, that is, we train model until convergence before the next step pruning. In this way, local network pruning increases computation consumption on local devices and delays the training. Recently, federated pruning has drawn much research attention. The models are kept being pruned together with the standard FedAvg learning process. Federated pruning allows us to train networks in a computation efficient as well as communication efficient manner because we only need to upload the non-zero parameters for synchronization. One potential drawback is that it is hard to find optimal pruning ratio.

It is worth mentioning that all the efficient approaches in both cross-silo and cross-device edge federated learning could be corrupted due to the system level heterogeneity and statistical level heterogeneity. System heterogeneity refers to the different hardwares (CPU, GPU, memory), network configurations, and power supplies of nodes in edge federated learning. Different computation capabilities may cause the unfairness results among local models and downgrade the fusion model. Different network configurations may cause important local model shard missing and increase training time. The statistical heterogeneity refers to the highly non-i.i.d training data. Nodes frequently collect and process data in a non-i.i.d manner. This against a commonly used assumption that all training data is drawn from an independent and identically distributed data source. The presence of non-i.i.d data used in edge federated learning leads to local models divergence. The further network quantization and pruning makes the divergence problem even worse. In the following sections, we will introduce communication efficient and computation efficient techniques.

### *4.2. Communication Efficient Methods*

The optimization methods for federated learning are largely inheriting from the conventional distributed machine learning optimization. The distributed first-order stochastic gradient descent (SGD) optimization methods have been largely studied in literature [29, 30, 31, 32, 33]. Local-SGD, as another approach to train the neural network in a distributed manner with less communication has been studied in [34]. They considered a master-worker topology and provided theoretical analysis for the convergence of local-SGD. The fundamental difference between distributed SGD and local-SGD is the use of training data. More specifically, if every local node uses training data that comes from the same data distribution, the local-SGD is equivalent to its distributed version. However, if local nodes use arbitrarily heterogeneous training data, local-SGD and distributed SGD are entirely different. We cannot expect the model updates (or gradients) to be drawn from the same unknown distribution even when local epoch $e = 1$. Although we cannot directly use distributed version techniques to address the communication bottleneck in edge federated learning, the vector quantization and sparsification are still main-stream optimization strategies for edge federated learning. In this section, we summarize some methods of communication efficient training in conventional distributed learning and discuss how they related to edge federated learning.

#### *4.2.1. Gradient Qantization*

When using SGD or other first order gradient method as model optimizer, quantizing the gradients to its low precision value has been widely adopted. Gradient quantization has been explored in the [35, 36, 37, 38, 39]. In particular, [40] summarized the general gradient quantization scheme $Q(g, \mathbf{s}, l)$ as

$$\hat{g}_t = \mathbf{s} \cdot \mathrm{sgn}(g_t) \cdot \kappa(g_t, l) \tag{4}$$

where $\mathbf{s}$ is a shared scaling factor (possible choices include $\|g_t\|_2$ or $\|g_t\|_\infty$), and $\text{sgn}(\cdot)$ returns the sign of gradient coordinate $g_t$. $\kappa(\cdot,\cdot)$ is an independent random variable defined as follows. Let $0 \leq k \leq l$ be an integer such that $|g_t|/\|g_t\| \in [p/l, (p+1)/l]$, then

$$\kappa(g_t, l) \triangleq \begin{cases} p/l, & \text{w.p } p - \frac{|g_t|}{\mathbf{s}} \cdot l + 1 \\ (p+1)/l, & \text{otherwise} \end{cases} \tag{5}$$

A concrete explanation of the above formula is from [36]. TernGrad compresses gradients into ternary values $\{-1, 0, 1\}$ with a stochastic quantization function to ensure the unbiasedness. Terngrad sets the quantization level $l = \mathbf{s}$, and chooses shared scaling factor among all workers such that $\mathbf{s} = \max(\|g_t\|_\infty)$ for all $m \in [N]$. Other work such as [35], the authors adventurously applied 1-bit SGD on speech DNNs to reduce data-exchange bandwidth and they empirically showed its feasibility on distributed environments. An error feedback scheme is introduced during quantization, to compensate for the quantization error. [38] proposed the DoReFa-Net to train convolutional networks with weights, and gradients all quantized into fixed-point numbers.

FedPAQ [41], to our best knowledge, maybe the first study that bridges the gap between distributed gradient quantization and federated learning. The idea of FedPAQ is quite straightforward: using quantized model updates during the FedAvg process. Applying gradient quantization methods to federated model updation should be very careful. The model divergence is enlarged by training on highly distributed non-i.i.d data. Especially, when the training scale increases, quantized model updates introduce lots of quantization variances. One possible solution is to use error compensation quantization for model updation mentioned in [42]. The variance reduced SGD [43] and variance reduced quantized SGD [44] are also helpful.

### *4.2.2. Gradient Sparsification*

Gradient sparsification is also a popular and efficient method. The intuition of sparse approaches is straightforward, that is, dropping the less beneficial coordinates of gradient vectors and then synchronizing on PS to ensure the unbiasedness. Therefore, reduction of communication cost is adopted. The insight of gradient sparsification can attribute to:

- DNNs usually over-parameterized [45] with a considerable number of parameters that have numerical values close to zero, and therefore resulting in spare sub-gradients.

- Sparse SGD can be formulated as variants of delaying weight updates such as asynchronous SGD [46].

To avoid the information loss when coordinate dropping occurs, [47] applied gradients accumulation for gradient value less than the predefined updating threshold. Unfortunately, updating those out-of-data (stale) gradients cause convergence, slow down and model performance degeneration. Early studies such as [48, 49, 50, 51] used similar constant-like or fixed ratio thresholds to perform gradient coordinates selection. It is impractical to use the above methods, because the threshold is hard to choose for particular DNNs and our experiments show the threshold-based methods even fail to converge in some cases. Recently, a series of hybrid strategies via combining gradient sparsity and vector quantization have been proposed. A heuristic algorithm proposed by [52] aimed to automatically tune the compression rate and then quantize them for updates. With limited coding length of stochastic gradients, and constrained gradient variance budget, [53] achieved a high compression ratio on $l_2$-regularized logistic regression by using gradient sparsification technique.

The gradient level sparsification training possibly leads the final model to be sparse. It will accelerate the model training on nodes later. One tightly related work to gradient sparsification is [54]. The method so called Federated Drop aims to train the randomly selected sub-model. These sub-models are subsets of the global model and, as such, the computed local updates have a natural interpretation as updates to the larger global model. Sparsification is an easily applied method. It does not require any network topology changes or extra computation bandwidth. However, the gradient selection is challenging and non-trivial and yet this is still an open question in edge federated learning.

### *4.3. Computational Efficient Methods*

Deep neural networks have made significant improvements in lots of computer vision tasks such as image recognition and objective detection. This motivates interests to deploy the state-of-the-art deep models to real world applications like mobile devices. For those applications, it is typically assumed that training is performed on the server and test is executed on mobile devices. However, in the cross-device edge federated learning scheme, both training and inference phases are located on mobile devices. These models often need considerable storage and computational power, and can easily overburden the limited storage, battery power, and computer capabilities of the model devices.

#### *4.3.1. Network Quantization*

To address the computational and storage issues, methods using quantized weights or activations in models have been proposed. The network has been accelerated by quantizing each full-precision weight to a small number of bits. This can be further divided to two sub-categories, depending on whether approximating full-precision weights with the linear combination of multiple binary weight bases at each iteration [55, 56, 57, 26, 58] or the model loss information is used [59, 60, 61]. The former uses different weight quantization resolution such as binary weights [62], which uses only one bit for each weight while still achieving state-of-the-art classification results. Also [26] and [63] added scaling to the ternarized weights, and DoReFa-Net [38] further extended quantization to any quantization levels. In a weight quantized network, $m$ bits where $m \geq 2$ are used to represent each weight. Let $\mathcal{Q}$ be a set of $2k+1$ quantized values, where $k = 2^{m-1}+1$. The linear quantization scheme has $\mathcal{Q} = \{-1, -\frac{k-1}{k}, \cdots, \frac{k-1}{k}, 1\}$ and logarithmic quantization scheme has $Q = \{-1, -\frac{1}{2}, \cdots, -\frac{1}{2^k-1}, 0, \frac{1}{2^k-1}, \cdots, 1\}$. When $m = 2$, both schemes reduce to $\mathcal{Q} = \{-1, 0, 1\}$. Both quantization schemes can be applied to any hidden layer in the model. Particularly, in order to constrain a CNN to have binary weights, a series of binary filters $B_1, B_2, \cdots, B_n \in \{-1, 1\}^{c_{in} \times w \times h \times c_{out}}$ is used to estimate the real-value weight filter $W \in R^{c_{in} \times w \times h \times c_{out}}$ such that $W \approx \alpha_1 B_1 + \alpha_2 B_2 + \cdots \alpha_n B_n$ which is a linear combination of $n$ binary or tenary filters. Here $c_{in} \times w \times h \times c_{out}$ is the dimension of weights. The optimal estimation can be solved by minimizing the following optimization problem.

$$\min J(\alpha, B) = \|W - \alpha B\|^2, \tag{6}$$

In addition, another approach known as loss-aware network quantization minimizes the loss directly w.r.t the quantized weights and often achieves better performance than approximation-based methods. The existing weight quantization methods above simply find the closest approximation of weight and ignore its effects to the model loss. However, it uses full-precision weights during the training process and extra gradient information, which is expensive [59].

In edge federated learning, if we could perform training with a quantized model on each device, it can significantly reduce the computational burden and accelerate the training and inference. To our best knowledge, there is no such work that can fully adapt to the edge federated learning environment. One close approach [28] tried to update the global model by using quantized local modes. It reduces the communication cost, however the local computational cost increases because it uses a full-precision model for local training and extra work on computing ready-update quantized models. Most of the aforementioned network quantization methods cannot be directly used in edge federated learning without modification. The challenge is we cannot ignore the model divergence in edge federated learning and inappropriate quantization introduces much mode noises to the global model, which makes convergence speed slow.

### *4.3.2. Network Pruning*

Neural network pruning is an alternative way to reduce the complexity of neural network models and accelerate the deep neural network on resource-limited edge nodes. Continuously dropping the small magnitudes weights and finding an optimal substructure of the original network is the key mechanism of pruning methods. It can be well explained by lottery ticket hypothesis [64]. Magnitude-based pruning methods including [65, 66, 67, 68, 69] that train until convergence before the next pruning step is prohibited on edge nodes. The iterative pruning methods [70, 71] is more attractive. The dynamic pruning allows the network to grow and shrink during the training. These existing pruning techniques consider the centralized setting with full access to the training data, which is fundamentally different from edge federated learning settings. The pruning method for decentralized data training is under discussion. PrunEdge FL [72] proposed a two-stage distributed pruning algorithm for federated learning. At beginning, a shared pruning model is sent to each node to train. Then

PrunEdge FL performs dynamic pruning together with the standard FedAvg procedure. One drawback of magnitude-based and adaptive pruning methods is that it is difficult to control the model size for the update. The structure of the submodel constantly changes over the training.

### *4.4. Other Efficient Methods*

Straggler problem, where the nodes lag because of computational resource heterogeneity recently has drawn much attention [73, 41, 74]. Synchronously or asynchronously updating models over heterogeneous network configurations is quite challenging. A tier-based federated learning framework that updates local model parameters synchronously within tiers and updates the global model asynchronously across tiers are proposed by [73]. Another approach is called HeteroFL [75]. By coordinatively training local models which are smaller than a global model to produce a single global inference model, HeteroFL is robust against the non-i.i.d statistical heterogeneity.

Optimal sampling problem is another interesting problem. The number of nodes in edge federated learning is in the order of millions and each node in the system has very limited contribution to the global model for each round. By sampling the important nodes, we can tremendously save communication costs and accelerate the training process. The [76] uses Ornstein-Uhlenbeck (OU) process, a continuous stochastic process to adaptively decide the node-side model updation. [77] uses a non-uniform sampling scheme, where the nodes are sampled according to some predefined distribution.

### *4.5. Future Directions*

Communication and computation are the key bottlenecks to consider when developing methods for edge federated networks. Using traditional methods such as gradient quantization and sparsification is less beneficial as we discuss early. The current efficient method studies are based on the standard FedAvg process and its variants. It is necessary to discover more efficient algorithms other than FedAvg which are more suitable for Federated learning schemes. There are some conditions that new algorithms should satisfy:

- It can achieve the same convergence speed as FedAvg and at least have the same performance as FedAvg.
- It can deal with both systems heterogeneity and statistical heterogeneity challenges in edge federated learning.
- It can be easily applied to any edge federated learning applications (image recognition, NLP, etc.).

Alternatively, the communication efficiency can be reduced through fast model training (using less communication rounds) or important node sampling (using less nodes) based on equation 3.

Computation efficiency, on the other hand, is another bottleneck for edge federated learning development. The core idea is to reduce the workload of local nodes by using lightweight models to reduce the usage of computation resources and memory. One possible solution is we could use neural architecture search (NAS) to find the optimal model for nodes. Besides NAS, we can also use split learning [78]. Many machine learning tasks have a heavy computation on the fully connected layers. We can transfer these computation burdens to a powerful edge server. The training data is processed on the local node therefore privacy is still preserved.

## 5. Security and Privacy

Security and privacy problems are two major problems of implementing federated learning in edge computing. Because of the natural heterogeneous environment of federated learning and edge computing, it is always hard to predict activities of other nodes in the system. For example, in the training process of edge federated learning, due to the uncertainty of other nodes, some nodes may be malicious and attack the training process. In addition, the curious nodes and servers may find it interesting to learn our personal data and want to retrieve the private data from the update information we upload in each synchronization. In this scenario, the security and privacy of our node may be harmed. In summary, the security and privacy problems are divided into two parts: how to attack and how to defend. In this section, we will introduce the current attack and defense algorithms in edge federated learning.

### *5.1. Security in Edge Federated Learning*

Security issues arise in edge federated learning because of the heterogeneity of both edge computing and federated learning. On the one hand, edge nodes and edge servers are usually from different sources while at the same time, not all of them are trusted. On the other hand, in federated learning, we usually assume that all nodes keep their own private data and do not share them with other nodes or servers. Those features improve the application generality of edge computing and keep more privacy for users' private data, but they also increase the risk of suffering malicious attacks. Much work has been done to address security issues in edge federated learning. We will first introduce two major ways to inject attacks followed by some existing algorithms to defend attacks.

### *5.1.1. Attacks*

In this section, we introduce two major security attacks in edge federated learning: Byzantine attack and poisoning attack.

*Byzantine attack.* Byzantine problems have been explored since the beginning of the distributed system. This problem was first introduced by Lamport et al. [79] in 1982 and talks about the fails of the whole distributed computing systems if some nodes are attacked, compromised or failed. This problem was first introduced into the distributed machine learning area by Blanchard et al. [80] in 2017. We briefly define this problem together with Figure 2, which is the structure of the federated learning system. As this figure shows, in the synchronous federated learning, Byzantine problems exist when some nodes are attacked or compromised and do not compute or upload weights correctly. In this scenario, the uploaded weights $w_t^i$ in (2) may not be the real $w_t^i$ computed by (1). Theoretically, the generalized Byzantine model that is defined in [80, 81] is:

**Definition 1 (Generalized Byzantine Model).**

$$w_t^{i,(r)} = \begin{cases} w_t^i & \textit{if i-th node is honest} \\ a_i \neq w_t^i & \textit{otherwise} \end{cases} \tag{7}$$

Here we denote $w_t^{i,(r)}$ as the actual gradient received by central server from $node_i$. In each iteration of the training phase, some nodes may become Byzantine nodes and upload an attack gradient $a_i$ to the server. As we can see in Figure 2, $node_3$ here is a Byzantine attacker. It uploads an alternative $w_t^{3,(r)}$ rather than the actual $w_t^3$ to the server. The central server, at the same time, does not know $node_3$ is compromised. It aggregates all the uploaded gradients and sends the incorrect updated weights back to all nodes. According to Theorem 1 in [82], when the aggregation function is an average function, one Byzantine attacker can take over the aggregation result and lead the whole training process to an incorrect phase.

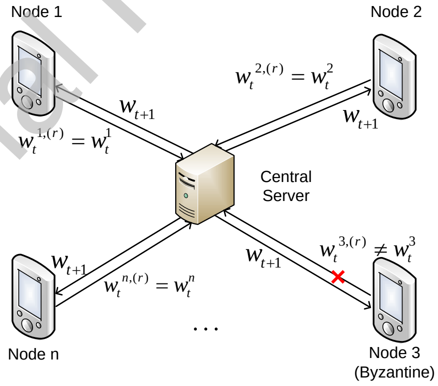

Figure 2: Federated learning with Byzantine attackers

As we said before, federated learning is a special kind of distributed machine learning, so the Byzantine problems remain the same or get even worse in edge federated learning. The main differences are listed below:

- Edge computing environment is complex, where edge nodes and servers are usually from different sources. This increases the possibility of suffering Byzantine attacks in edge federated learning.

- In the federated learning, the data on each node is private and does not share with each other, so the distribution of subdataset on each node can be either non-i.i.d. or i.i.d., while the heterogeneity makes it easier to attack and harder to detect.
- In each iteration of the federated learning, only some of nodes are selected to perform the computation, making the honest majority assumption unreasonable. It is possible that more than half of nodes are under attacks in each iteration.

Mainly there are three different ways to inject Byzantine attacks [81].

- Gaussian attack. Simply generate Gaussian noise as attack gradients and weights.
- Omniscient Attack. Let the direction of gradients uploaded by the Byzantine nodes equal to the direction of the sum of all honest nodes.
- Flip bit attack. Flip some bit of the uploaded gradients and weights.

*Poisoning attack.* Poisoning attack is a kind of attack to injection attacks in the federated learning training process. In general, there are two types of poisoning attacks: data poisoning and model poisoning.

**Data Poisoning** Data poisoning is a naive way to attack the federated learning system. It uses a simple idea: affecting the training process by changing the input data. One method to inject data poisoning attacks is proposed by Tolpegin et al. [83]. Their method is based on a simple intuition by flipping the labels [84] of the input data. The labels of the input data are randomly shuffled such that the input data mismatch their corresponding labels, which has a significantly negative impact on the classes that are under attack. Besides, Shafahi et al. [85] propose a clean-labels attack. They introduce an optimization-based method for crafting poisons without requiring the attackers to make any modifications on the input data label. By conducting this attack, they can make the model fail at some specific task. Gu et al. choose to mix the clean data with adversarial data [86] to attack the model training [87].

**Model Poisoning** Unlike data poisoning, model poisoning does not manipulate the input data. It aims at manipulating the local model to inject backdoors or malicious attacks into the global model. Bagdasaryan et al. first introduced backdoor attack into federated learning area [88]. They use a model replacement method to inject this attack from one or multiple compromised nodes. Therefore, after aggregation in the central server, the updated global model will be injected and misclassify some predefined inference tasks. Bhagoji et al. proposed another method to inject the targeted model poisoning and stealthy model poisoning for standard federated learning [89]. They choose the attack gradients and weights by estimating the benign nodes updates and optimizing for both the training loss and adversarial objective. Their experiments show a high successful attack rate against some Byzantine-resilient algorithms such as Krum [80] and coordinate-wise median [90]. Fang et al. focus on attacking the model training with four different Byzantine robust algorithms [91]. In order to attack those algorithms, they solve an optimization problem for each objective algorithm.

### *5.1.2. Defenses*

Although the methods to attack edge federated learning are different for Byzantine attacks and poisoning attacks, the defense problems are actually from the same structure, that is, the central server must distinguish the information uploaded by honest nodes from the information uploaded by attack nodes. In the related literature, all those defense algorithms are denoted as Byzantine-resilient algorithms.

To defend Byzantine attacks in distributed machine learning, there are basically three different directions.

- Score-based method. This direction usually defines a metric to score each uploaded weight and finally we can choose the one with the highest score as the aggregation results.
- Median-based method. Geometric median and its modifications are used in this direction.
- Distance-based method. This kind of method uses the distance information in euclidean space to remove outliers.

Blanchard et al. first proposed a score-based algorithm called Krum [80] to measure the scores for each uploaded gradient in the central server by the L2 norm sum of its closest $n - f - 2$ gradients and chose the gradient with the highest score as the aggregated gradient. After this, they also proposed another algorithm to resist asynchronous Byzantine attacks [92]. However, Krum has a natural shortcoming that they can only choose one gradient among all uploaded gradients, which reduces the convergence speed a lot.

As for median-based methods, in fact, most of the following work usually focused on using median-based aggregation methods rather than predefined score-based methods. For example, Xie et al. proposed geometric median, marginal median and median-around-median [81], Yin et al. proposed coordinate-wised median [90], Su et al. proposed a batch normalized median [93], Alistarh et al. proposed a more complicated modification of median-based methods that is called ByzantineSGD [94]. However, because the geometric median is a point that minimizes the sum of distances to all points, in order to find the geometric median, a recursive method is adopted and therefore the time complexity is really large.

The last direction is distance-based method. Yin et al proposed coordinate-wise trimmed mean [90], Xia et al. proposed an alternative method called FABA [82]. Instead of using geometric median to aggregate the uploaded gradients, those methods used euclidean distance to remove outlier gradients. They adaptively remove outliers based on the center current remaining gradients. They later provided another Byzantine-resilient algorithm for large scale distributed machine learning [95].

As for the federated learning area, several Byzantine robust algorithms are proposed. The biggest difference between federated learning and classic distributed machine learning is that the subdataset on each node is non-i.i.d. distributed. Although all those algorithms in classic distributed machine learning methods still work in some scenarios, they really depend on how non-i.i.d. the datasets are. Ghosh et al. first talked about this problem in 2019 [96], in which they combine K-means and trimmed mean together to achieve Byzantine-resilient. They use K-means to gather the uploaded weights into several clusters and use trimmed-means to remove the outliers. However, there is one problem here that maybe all uploaded weights in the same cluster are from Byzantine nodes, and thus may affect the performance. Muoz-Gonzlez et al. proposed an adaptive model averaging algorithm to resist Byzantine attacks [97]. They divide all nodes into two sets: good clients set and bad clients' set. Then in each iteration, they compare the uploaded weights with the aggregation results from nodes in good clients' set and update both sets. Then they use the updated good clients' sets to perform the aggregation in the current iteration. Prakash et al. proposed a method based on the direction similarity and length similarity [98]. Kang et al. proposed a decentralized method to achieve reliable federated learning for mobile networks [99].

### *5.1.3. Future Directions*

Right now, all the proposed defense methods assume that either the data distribution is i.i.d. on all nodes or the majority of the nodes are honest. However, these assumptions are not practical in edge federated learning. First, edge nodes usually collect their data from their own sources, so the data distribution is not necessarily i.i.d. Second, at synchronization time, the central server randomly selects some of the edge nodes to perform the computation, among which the honest nodes may not be the majority even if more than half nodes are honest. In this setting, honest majority is not a reasonable assumption through the whole training process.

Therefore, a more practical algorithm without assuming i.i.d. data distribution and honest majority is expected in the future to defend the security attacks to edge federated learning. At present, there are very few explorations in this direction. How to perfectly defend Byzantine attacks remains an open problem.

## *5.2. Privacy in Edge Federated Learning*

Although federated learning is designed to protect each node's private training data without relying on training data transmission between servers, privacy breach can still be incurred when information (e.g., model weights) is shared between servers. It is possible that some curious nodes or servers can extract private information through the training process. In this subsection, we will introduce various privacy attacks and privacy-preserving algorithms.

*5.2.1. Attacks*

In edge federated learning, the information exchange between nodes and server basically contains weights that are computed by nodes using local private data and the updated weights that are aggregated by the central server. Therefore, in order to attack the privacy of some nodes, the process of the adversary is to retrieve the private dataset information from uploaded weights (curious server) or aggregated weights (curious node) [100]. Thus, this problem is equivalent to retrieving the data from weight update. In general, there are two types of privacy attacks in federated learning area.

- Membership inference attack [101]. This kind of attack is to determine whether a data record is contained in a node's training dataset. When the dataset is sensitive, this attack may leak a lot of useful information.
- Data inference attack. This attack aims to retrieve the training data or a class of training data from the information that node provides.

There is some existing work for both attack types, and we list some below.

Nasr et al. proposed a white-box inference attack in federated learning [102] and provided a comprehensive privacy analysis of deep learning models. Truex et al. proposed a feasible black-box membership inference attack in federated learning [103]. Zhu et al. proposed a method called deep leakage to retrieve training data from public shared gradients on both computer vision and natural language processing tasks [104]. Their method is based on minimizing the loss between the dummy gradients computed by the attack training data and real gradients computed by the true training data. Experiments show a very large leakage rate for four different datasets. This shows that a curious server can easily retrieve the training data by the gradients that node uploads. Hitaj et al. proposed an information leakage method [105] using generative adversarial networks [106] in collaborative deep learning. Although this needs a separate neural network to retrieve the training data, it has an excellent performance in information leakage even with privacy-preserving algorithms. Wang et al. use a similar GAN-based method called Multi-task GAN in federated learning [107] to precisely recover the private data from a specific client which causes user-level privacy leakage.

*5.2.2. Defenses*

The privacy-preserving techniques in edge federated learning are roughly in two directions.

- Algorithm-based solutions. The most widely used technique in the machine learning area to protect privacy is differential privacy [108]. The common technique is to add noise while maintaining an acceptable performance.
- Encryption-based solutions. These kinds of solutions are in low-level architecture to encrypt the communication information to protect privacy such as secure multi-party computation [109].

Most of the existing algorithms are in these two directions, while some of them are the combinations of several techniques to protect privacy. There is several work talking about implementing differential privacy techniques in federated learning area. Wei et al. proposed an algorithm called NbAFL, which adds artificial noise before aggregation [110]. Geyer et al. considered this problem in the client-side perspective and they use random sub-sampling and Gaussian mechanism to distort the sum of all updates [111]. Bhowmick et al. designed a new optimal locally differentially private algorithm for all privacy levels [112]. Ghazi et al. proposed a simple differential privacy-based algorithm for a shuffled model in federated learning to make user's data indistinguishable with random noise [113].

Additionally, in encryption level, Phong et al. used an additively homomorphic encryption in cryptography area in collaborative deep learning and built a privacy-preserving enhanced system [114]. When communicating with the central server, the information is well encrypted so that no information is leaked to the curious-but-honest server and accuracy is kept intact. Elgabli et al. proposed A-FADMM [115], which is based on analog transmissions and the alternating direction method of multipliers. This method can hide each local model's update trajectory from any eavesdropper, which protects privacy of each node.

Some hybrid methods are also proposed in this edge federated learning. Truex et al. proposed a hybrid method to implement differential privacy and secure multiparty computation at the same time [116]. This method reduces the noise injection into communication with the central server to increase the performance and still maintain a high privacy level. Hao et al. integrated additively homomorphic encryption [114] with differential privacy [117]. Their method provides a stronger protection to prevent privacy leakage in the scenario that multiple nodes or central servers are colluded. They also proposed another method by adding encryption-level and differential privacy protection in federated learning [118]. This method supports large-scale federated learning applications.

### *5.2.3. Future Directions*

Much work has been done in addressing the privacy issues in machine learning or federated learning, in particular, how to solve information leakage when transmitting gradients and weights between nodes and the central server. Little work has focused on privacy issues in edge federated learning. The privacy issue, however, may become more severe in edge computing because edge nodes/servers may leak information regarding data, usage, location to malicious users [119, 120]. How to preserve privacy in edge federated learning by considering the specifics of edge computing is a new direction worth exploration.

## 6. Migration and Scheduling

Migration and scheduling are two major low-level supports in edge computing [121, 122]. In this section, we will compare their differences between edge computing and edge federated learning.

### *6.1. Migration*

In edge federated learning, the migration problem arises when the edge node moves between edge servers. Because we know that in edge computing, edge nodes are always connected to the geographically nearest edge server to get low latency and high bandwidth, it is possible for the edge node to travel from one edge server to another while in the process of federated learning training. In this scenario, the new connected edge server does not have a copy of the federated learning model and thus a migration between edge servers must be implemented. From Section 4, we know that a mid-size model can be around hundreds of megabyte level, so it takes time to migrate the model between edge servers, especially when we adopt split learning [123, 124] to offload most of the computation on edge servers. Therefore, a collaborative and efficient migration policy is necessary in edge federated learning.

There is a lot of research about migration in edge computing [121, 125, 126, 127]. However, in edge federated learning area, research about migration is still at an early stage. Although it is possible to let the under-layer infrastructure handle the migration process between different edge server, it is too heavy to migrate the model in system level. Because the federated learning training has some extra features such as large model size, flexible node selection, we should specifically optimize the migration process for edge federated learning itself.

We believe the following strategies may help to further improve the migration-based edge federated learning performance.

- A naive way is simply to keep a copy in all the adjacent edge servers. This method is fairly effective and achieves very low latency. However, this process has a huge communication and storage cost.
- We can also split the network into several parts and broadcast part of the model to other edge servers, this can reduce the communication cost in the migration process and stay a low storage cost.
- It is efficient to predict the movement path of the edge nodes using a separate model and cache the model to the predicted edge servers for a better migration experience.
- If the migration cost is too high, we can update the scheduling policy by abandoning the computation results from this node and reuse it after finishing the migration process.

### *6.2. Scheduling*

Scheduling, or resource allocation is an important problem in edge computing. Due to the node and server heterogeneity, the data, computation, memory, and network resources vary a lot among different devices. In federated learning, a synchronous iteration requires all participating nodes finish their computation and upload computational results to the central server before the server continues to perform the aggregation and model update. Therefore, the training speed is limited by the node with slowest computational resources and network bandwidth. In order to get a better training speed, it is better to schedule the computational resource in an efficient way.

*6.2.1. Current Solutions*

There is much previous work about scheduling and resource allocation in edge federated learning area. In summary, most of the solutions follow the following four directions.

- Participant selection. In federated learning, the central server randomly selects some nodes to perform the computation. Therefore, it helps efficiency to select the participated nodes in a smarter way.
- Resource optimization. In edge computing, because of the heterogeneity of nodes, the computational and network resources in each device are different. We can optimize the resource allocation by letting nodes with more computational power compute more.
- Asynchronous training. Most of the current edge federated research focus on synchronous training, but asynchronous training can significantly improve the efficiency in a heterogeneous environment.
- Incentive Mechanism. Some of the researchers focus on the incentive compensation in federated learning because the node must consume their computational resources for collaborative work. An efficient incentive mechanism can help invite more participants while maximizing the usage of the resources.

We list some solutions based on these four directions below.

*Participant Selection.* The idea of participant selection is based on the mechanism of federated learning. In federated learning, some nodes are randomly selected as participants to perform the computation using their private local data for one iteration, among which some may have enough computational resources and high network bandwidth, and some may have limited resources. The one with the slowest computational speed and upload speed decides the total training time for this iteration. Therefore, it is very useful to choose the participating nodes in a smarter way. Nishio et al. proposed a novel federated learning protocol FedCS to mitigate this problem [128]. The main part of FedCS is a client selection protocol. They initialize the whole framework by requesting the resource information of all nodes. Then in order to select the clients, they solve an optimization problem by the computational resource and previous training time information. This method is efficient in client selection but may affect the performance because of the non-i.i.d. distributed data. Yoshida et al. later proposed an enhanced framework called Hybrid-FL [129]. Similar to their previous work, they also request the resource information of all nodes at first. However, they choose to perform a client and data selection at the same time instead of just selecting the clients to decrease the influence of the data distribution. Then they upload the selected data and update the model. Yang et al. developed a formal framework to analyze different scheduling policies' convergence performance [130]. Through their analysis, using proportional fair scheduling policy to select clients performs better than randomly scheduling and round robin.

*Resource Optimization.* Most of the proposed methods for resource allocation transform this problem to a resource optimization problem, that is, given constraints about the edge node computation resources and network limit, the goal is to find the most efficient way to implement the edge federated learning. For example, Dinh et al. proposed FEDL and solved an optimization problem to minimize energy and time consumption [131]. Li et al. proposed q-FedAvg optimization objective for fair resource allocation in edge federated learning [132]. Zeng et al. proposed energy-efficient strategies for bandwidth allocation and scheduling [133]. Neely et al. proposed a scheduling control of heterogeneous network for resource allocation [134]. Similar algorithms are also proposed in [135, 136, 137]. Apart from optimization, Zou et al. considered about using game theory for resource allocation [138]. They proposed an evolutionary game approach to dynamically schedule the computing resources and reach an evolutionary equilibrium. Recently there are some work about using reinforcement learning for scheduling the resources. Nguyen et al. proposed a deep reinforcement learning based method [139]. They use a neural network to decide the scheduling policy and update the network by the corresponding rewards. Zhan et al. also proposed a deep reinforcement learning based method [140] to get a near-optimal solution for the optimization problem without knowledge about the networks.

*Asynchronous training.* Asynchronous training fits well for resource allocation problems. It is because in asynchronous training, the central server does not have to wait for all the participating nodes to finish their computation before updating the global model. In this scenario, nodes can take their time to train on their private local data even if they lack computational resources or suffer the network delay. Chen et al. proposed ASO-fed, an asynchronous online edge federated learning framework [141]. Central server will take streaming of model updates from different edge nodes because of the node heterogeneity and update the model accordingly in an exponential moving average way with non-i.i.d. and imbalanced setting. Lu et al. combined differential privacy and asynchronous federated learning together to reach both privacy guarantee and better resource allocation [142]. Chen et al. used a temporally weighted aggregation method in the setting of asynchronous federated learning in order to make use of the previously trained local models [143]. This helps the convergence speed and accuracy performance. Chen et al. proposed a vertical asynchronous federated learning method VAFL by using a perturbed local embedding [144], which improves the data privacy and communication efficiency.

*Incentive Mechanism.* Incentive mechanism addresses how to reward the participating edge nodes according to their computational resources and personal data so that they are willing to contribute their computational power for a collaborative federated training. A practical incentive mechanism must be fair for both participated nodes and edge servers. Kang et al. introduced a contract theory based incentive design [12]. They use a contract model to define the data quality of edge nodes and give more reward to data owners who have high quality data. However, they only consider the price of data, but no price of computational resources. Feng et al. combined rewards for providing data and computation resources in one pricing model [145]. They use a Stackelberg game model to evaluate the value of edge nodes. Khan et al. also use a similar game model [146].

### *6.2.2. Future Directions*

We summarized three future directions for scheduling in edge federated learning. First, current algorithms for scheduling and resource allocation always try to minimize the training time. However, in this setting, the central server may not select nodes with limited computational resource or unstable network because of the long waiting time. The data on those nodes will not be used in the model training, which results in a biased model training. To mitigate this problem, we may group the nodes with similar training time together and integrate their weights before sending them to the server in batches. This way, all data can be used for training and the training time can be reduced as well. Second, in asynchronous training, most of the previous work focuses on using numerical experiments to show the performance. However, there is still a lack of theoretical study for asynchronous training. Mathematical analysis and comparisons between asynchronous training and synchronous training are needed in edge federated learning. Third, the incentive mechanisms in edge federated learning seem to be a less studied topic. It is promising to explore how to incentivize the participation of nodes with high-quality data and abundant computational resources.

## 7. Conclusion

In this article, we carefully investigate edge federated learning, which is a paradigm to implement federated learning on edge computing environments. The development of edge federated learning is still at an early stage, and there is not much research in this area. We summarize the research problems and methods respectively in applications, development tools, communication efficiency, security, privacy, migration and scheduling as well as providing some insights of the future directions and open problems in edge federated learning. With the fast advancement of both edge computing and federated learning, more and more collaborative training methods for edge federated learning are developed for better user experience and privacy protection. We will need more efforts on solving those open problems in edge federated learning.

## References

[1] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39. doi:10.1109/MC.2017.9.

[2] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, Future Generation Computer Systems 97 (2019) 219–235.

[3] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Processing Magazine 37 (3) (2020) 50–60.

[4] B. Liu, B. Yan, Y. Zhou, Y. Yang, Y. Zhang, Experiments of federated learning for covid-19 chest x-ray images, arXiv preprint arXiv:2007.05592.

[5] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, S. Bakas, Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation, in: International MICCAI Brainlesion Workshop, Springer, 2018, pp. 92–104.

[6] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, et al., Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data, Scientific reports 10 (1) (2020) 1–12.

[7] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, Journal of Healthcare Informatics Research (2020) 1–19.

[8] Y. Chen, X. Qin, J. Wang, C. Yu, W. Gao, Fedhealth: A federated transfer learning framework for wearable healthcare, IEEE Intelligent Systems.

[9] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, S. Choudhary, Federated learning with personalization layers, arXiv preprint arXiv:1912.00818.

[10] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, S. Srikanteswara, Energy demand prediction with federated learning for electric vehicle networks, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[11] D. Ye, R. Yu, M. Pan, Z. Han, Federated learning in vehicular edge computing: A selective model aggregation approach, IEEE Access 8 (2020) 23920–23935.

[12] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, D. I. Kim, Incentive design for efficient federated learning in mobile networks: A contract theory approach, in: 2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS), IEEE, 2019, pp. 1–5.

[13] A. Imteaj, M. H. Amini, Distributed sensing using smart end-user devices: pathway to federated learning for autonomous iot, in: 2019 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2019, pp. 1156–1161.

[14] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, F. Beaufays, Applied federated learning: Improving google keyboard query suggestions, arXiv preprint arXiv:1812.02903.

[15] S. Ramaswamy, R. Mathews, K. Rao, F. Beaufays, Federated learning for emoji prediction in a mobile keyboard, arXiv preprint arXiv:1906.04329.

[16] F. Hartmann, S. Suh, A. Komarzewski, T. D. Smith, I. Segall, Federated learning for ranking browser history suggestions, arXiv preprint arXiv:1911.11807.

[17] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, 2013, pp. 15–20.

[18] N. K. Giang, M. Blackstock, R. Lea, V. C. Leung, Developing iot applications in the fog: A distributed dataflow approach, in: 2015 5th International Conference on the Internet of Things (IOT), IEEE, 2015, pp. 155–162.

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

[20] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, S. Zhao, Advances and open problems in federated learning, CoRR abs/1912.04977. `arXiv:1912.04977`.
URL `http://arxiv.org/abs/1912.04977`

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[22] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, arXiv preprint arXiv:1912.04977.

[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, When edge meets learning: Adaptive control for resource-constrained distributed machine learning, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 63–71.

[24] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[25] J. Ba, R. Caruana, Do deep nets really need to be deep?, Advances in neural information processing systems 27 (2014) 2654–2662.

[26] F. Li, B. Zhang, B. Liu, Ternary weight networks, arXiv preprint arXiv:1605.04711.

[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[28] M. M. Amiri, D. Gunduz, S. R. Kulkarni, H. V. Poor, Federated learning with quantized global model updates, arXiv preprint arXiv:2006.10672.

[29] B. Recht, C. Re, S. Wright, F. Niu, Hogwild: A lock-free approach to parallelizing stochastic gradient descent, in: J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 24, Curran Associates, Inc., 2011, pp. 693–701.
URL `http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf`

[30] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, A. Y. Ng, Large scale distributed deep networks, in: NIPS, 2012, pp. 1–2.

[31] T. M. Chilimbi, Y. Suzue, J. Apacible, K. Kalyanaraman, Project adam: Building an efficient and scalable deep learning training system, in: OSDI, 2014, pp. 1–2.

[32] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, CoRR abs/1512.01274. `arXiv:1512.01274`.
URL `http://arxiv.org/abs/1512.01274`

[33] Y. Zhang, J. Duchi, M. I. Jordan, M. J. Wainwright, Information-theoretic lower bounds for distributed statistical estimation with communication constraints, Advances in Neural Information Processing Systems 26 (2013) 2328–2336.

[34] S. U. Stich, Local sgd converges fast and communicates little, International Conference on Learning Representations.

[35] F. Seide, H. Fu, J. Droppo, G. Li, D. Yu, 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns, in: Interspeech 2014, interspeech 2014 Edition, 2014, pp. 1–2.
URL `https://www.microsoft.com/en-us/research/publication/1-bit-stochastic-gradient-descent-and-application-to-data-parallel-distributed-training-of-speech-dnns/`

[36] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, H. Li, Terngrad: Ternary gradients to reduce communication in distributed deep learning, CoRR abs/1705.07878. `arXiv:1705.07878`.
URL `http://arxiv.org/abs/1705.07878`

[37] H. Tang, C. Yu, X. Lian, T. Zhang, J. Liu, Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression, in: International Conference on Machine Learning, PMLR, 2019, pp. 6155–6165.

[38] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, arXiv preprint arXiv:1606.06160.

[39] J. Sun, T. Chen, G. Giannakis, Z. Yang, Communication-efficient distributed learning via lazily aggregated quantized gradients, Advances in Neural Information Processing Systems 32 (2019) 3370–3380.

[40] D. Alistarh, D. Grubic, J. Li, R. Tomioka, M. Vojnovic, Qsgd: Communication-efficient sgd via gradient quantization and encoding, Advances in Neural Information Processing Systems 30 (2017) 1709–1720.

[41] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, R. Pedarsani, Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2021–2031.

[42] J. Wu, W. Huang, J. Huang, T. Zhang, Error compensated quantized sgd and its applications to large-scale distributed optimization, arXiv preprint arXiv:1806.08054.

[43] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, Advances in neural information processing systems 26 (2013) 315–323.

[44] V. Gandikota, D. Kane, R. K. Maity, A. Mazumdar, vqsgd: Vector quantized stochastic gradient descent, arXiv preprint arXiv:1911.07971.

[45] A. Brutzkus, A. Globerson, E. Malach, S. Shalev-Shwartz, SGD learns over-parameterized networks that provably generalize on linearly separable data, in: International Conference on Learning Representations, 2018, pp. 1–2.
URL https://openreview.net/forum?id=rJ33wwxRb

[46] N. Strom, Scalable distributed dnn training using commodity gpu cloud computing, in: INTERSPEECH, 2015, pp. 1–2.

[47] Y. Lin, S. Han, H. Mao, Y. Wang, W. J. Dally, 5: Reducing the communication bandwidth for distributed training, CoRR abs/1712.01887. arXiv:1712.01887.
URL http://arxiv.org/abs/1712.01887

[48] R. Garg, R. Khandekar, Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property, in: Proceedings of the 26th International Conference On Machine Learning, ICML 2009, 2009, p. 43. doi:10.1145/1553374.1553417.

[49] N. Dryden, T. Moon, S. A. Jacobs, B. V. Essen, Communication quantization for data-parallel training of deep neural networks, 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC) (2016) 1–8.

[50] A. F. Aji, K. Heafield, Sparse communication for distributed gradient descent, CoRR abs/1704.05021. arXiv:1704.05021.
URL http://arxiv.org/abs/1704.05021

[51] Z. Tao, Q. Li, esgd: Communication efficient distributed deep learning on the edge, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), USENIX Association, Boston, MA, 2018, pp. 1–2.
URL https://www.usenix.org/conference/hotedge18/presentation/tao

[52] C. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, K. Gopalakrishnan, Adacomp : Adaptive residual gradient compression for data-parallel distributed training, CoRR abs/1712.02679. arXiv:1712.02679.
URL http://arxiv.org/abs/1712.02679

[53] J. Wangni, J. Wang, J. Liu, T. Zhang, Gradient sparsification for communication-efficient distributed optimization, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31, Curran Associates, Inc., 2018, pp. 1299–1309.
URL http://papers.nips.cc/paper/7405-gradient-sparsification-for-communication-efficient-distributed-optimization.pdf

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

[55] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, The Journal of Machine Learning Research 18 (1) (2017) 6869–6898.

[56] Z. Lin, M. Courbariaux, R. Memisevic, Y. Bengio, Neural networks with few multiplications, arXiv preprint arXiv:1510.03009.

[57] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: European conference on computer vision, Springer, 2016, pp. 525–542.

[58] X. Lin, C. Zhao, W. Pan, Towards accurate binary convolutional neural network, in: Advances in neural information processing systems, 2017, pp. 345–353.

[59] L. Hou, Q. Yao, J. T. Kwok, Loss-aware binarization of deep networks, arXiv preprint arXiv:1611.01600.

[60] L. Hou, J. T. Kwok, Loss-aware weight quantization of deep networks, arXiv preprint arXiv:1802.08635.

[61] C. Leng, H. Li, S. Zhu, R. Jin, Extremely low bit neural network: Squeeze the last bit out with admm, arXiv preprint arXiv:1707.09870.

[62] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Advances in neural information processing systems, 2015, pp. 3123–3131.

[63] C. Zhu, S. Han, H. Mao, W. J. Dally, Trained ternary quantization, arXiv preprint arXiv:1612.01064.

[64] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, arXiv preprint arXiv:1803.03635.

[65] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Advances in neural information processing systems 28 (2015) 1135–1143.

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

[67] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, Advances in neural information processing systems 29 (2016) 2074–2082.

[68] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710.

[69] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440.

[70] N. Lee, T. Ajanthan, P. H. Torr, Snip: Single-shot network pruning based on connection sensitivity, arXiv preprint arXiv:1810.02340.

[71] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, M. Jaggi, Dynamic model pruning with feedback, arXiv preprint arXiv:2006.07253.

[72] Y. Jiang, S. Wang, B. J. Ko, W.-H. Lee, L. Tassiulas, Model pruning enables efficient federated learning on edge devices, arXiv preprint arXiv:1909.12326.

[73] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, H. Rangwala, Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data, arXiv preprint arXiv:2010.05958.

[74] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Robust and communication-efficient federated learning from non-iid data, IEEE transactions on neural networks and learning systems.

[75] E. Diao, J. Ding, V. Tarokh, Heterofl: Computation and communication efficient federated learning for heterogeneous clients, arXiv preprint arXiv:2010.01264.

[76] M. Ribero, H. Vikalo, Communication-efficient federated learning via optimal client sampling, arXiv preprint arXiv:2007.15197.

[77] E. Rizk, S. Vlaski, A. H. Sayed, Federated learning under importance sampling, arXiv preprint arXiv:2012.07383.

[78] C. Thapa, M. A. P. Chamikara, S. Camtepe, Splitfed: When federated learning meets split learning, arXiv preprint arXiv:2004.12088.

[79] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, ACM Trans. Program. Lang. Syst. 4 (3) (1982) 382–401. doi:10.1145/357172.357176.
URL http://doi.acm.org/10.1145/357172.357176

[80] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 119–129.
URL http://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent.pdf

[81] C. Xie, O. Koyejo, I. Gupta, Generalized byzantine-tolerant SGD, CoRR abs/1802.10116. arXiv:1802.10116.
URL http://arxiv.org/abs/1802.10116

[82] Q. Xia, Z. Tao, Z. Hao, Q. Li, Faba: An algorithm for fast aggregation against byzantine attacks in distributed neural networks, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 4824–4830. doi:10.24963/ijcai.2019/670.
URL https://doi.org/10.24963/ijcai.2019/670

[83] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: L. Chen, N. Li, K. Liang, S. Schneider (Eds.), Computer Security – ESORICS 2020, Springer International Publishing, Cham, 2020, pp. 480–501.

[84] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: Proceedings of the 29th International Coference on International Conference on Machine Learning, ICML'12, Omnipress, Madison, WI, USA, 2012, p. 14671474.

[85] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, Poison frogs! targeted clean-label poisoning attacks on neural networks, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Curran Associates Inc., Red Hook, NY, USA, 2018, p. 61066116.

[86] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, J. D. Tygar, Adversarial machine learning, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 4358. doi:10.1145/2046684.2046692.
URL https://doi.org/10.1145/2046684.2046692

[87] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain (2019). arXiv:1708.06733.

[88] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: S. Chiappa, R. Calandra (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Vol. 108 of Proceedings of Machine Learning Research, PMLR, Online, 2020, pp. 2938–2948.
URL http://proceedings.mlr.press/v108/bagdasaryan20a.html

[89] A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR,

Long Beach, California, USA, 2019, pp. 634–643.
URL http://proceedings.mlr.press/v97/bhagoji19a.html

[90] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmssan, Stockholm Sweden, 2018, pp. 5650–5659.
URL http://proceedings.mlr.press/v80/yin18a.html

[91] M. Fang, X. Cao, J. Jia, N. Z. Gong, Local model poisoning attacks to byzantine-robust federated learning, in: S. Capkun, F. Roesner (Eds.), 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, USENIX Association, 2020, pp. 1605–1622.
URL https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[92] G. Damaskinos, E. M. El Mhamdi, R. Guerraoui, R. Patra, M. Taziki, Asynchronous Byzantine machine learning (the case of SGD), in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmssan, Stockholm Sweden, 2018, pp. 1145–1154.
URL http://proceedings.mlr.press/v80/damaskinos18a.html

[93] Y. Chen, L. Su, J. Xu, Distributed statistical machine learning in adversarial settings: Byzantine gradient descent, Proc. ACM Meas. Anal. Comput. Syst. 1 (2). doi:10.1145/3154503.
URL https://doi.org/10.1145/3154503

[94] D. Alistarh, Z. Allen-Zhu, J. Li, Byzantine stochastic gradient descent, CoRR abs/1803.08917. arXiv:1803.08917.
URL http://arxiv.org/abs/1803.08917

[95] Q. Xia, Z. Tao, Q. Li, Defenses against byzantine attacks in distributed deep neural networks, IEEE Transactions on Network Science and Engineering (2020) 1–1doi:10.1109/TNSE.2020.3035112.

[96] A. Ghosh, J. Hong, D. Yin, K. Ramchandran, Robust federated learning in a heterogeneous environment, CoRR abs/1906.06629. arXiv:1906.06629.
URL http://arxiv.org/abs/1906.06629

[97] L. Muoz-Gonzlez, K. T. Co, E. C. Lupu, Byzantine-robust federated machine learning through adaptive model averaging (2019). arXiv:1909.05125.

[98] S. Prakash, A. S. Avestimehr, Mitigating byzantine attacks in federated learning (2020). arXiv:2010.07541.

[99] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, M. Guizani, Reliable federated learning for mobile networks, IEEE Wireless Communications 27 (2) (2020) 72–80. doi:10.1109/MWC.001.1900119.

[100] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 691–706. doi:10.1109/SP.2019.00029.

[101] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 3–18. doi:10.1109/SP.2017.41.

[102] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 739–753. doi:10.1109/SP.2019.00065.

[103] S. Truex, L. Liu, M. Gursoy, L. Yu, W. Wei, Demystifying membership inference attacks in machine learning as a service, IEEE Transactions on Services Computing PP (2019) 1–1. doi:10.1109/TSC.2019.2897554.

[104] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, pp. 14774–14784.
URL https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

[105] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: Information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 603618. doi:10.1145/3133956.3134012.
URL https://doi.org/10.1145/3133956.3134012

[106] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 27, Curran Associates, Inc., 2014, pp. 2672–2680.
URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[107] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, H. Qi, Beyond inferring class representatives: User-level privacy leakage from federated learning, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2512–2520. doi:10.1109/INFOCOM.2019.8737416.

[108] C. Dwork, Differential privacy: A survey of results, in: M. Agrawal, D. Du, Z. Duan, A. Li (Eds.), Theory and Applications of Models of Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1–19.

[109] O. Goldreich, Secure multi-party computation, Manuscript. Preliminary Version.

[110] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, H. Vincent Poor, Federated learning with differential privacy: Algorithms and performance analysis, Trans. Info. For. Sec. 15 (2020) 34543469. doi:10.1109/TIFS.2020.2988575.
URL https://doi.org/10.1109/TIFS.2020.2988575

[111] R. C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective (2018). arXiv:1712.07557.

[112] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, R. Rogers, Protection against reconstruction and its applications in private federated learning (2019). arXiv:1812.00984.

[113] B. Ghazi, R. Pagh, A. Velingker, Scalable and differentially private distributed aggregation in the shuffled model, CoRR abs/1906.08320. arXiv:1906.08320.
URL http://arxiv.org/abs/1906.08320

[114] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Transactions on Information Forensics and Security 13 (5) (2018) 1333–1345. doi:10.1109/TIFS.2017.2787987.

[115] A. Elgabli, J. Park, C. B. Issaid, M. Bennis, Harnessing wireless channels for scalable and privacy-preserving federated learning (2020). arXiv:2007.01790.

[116] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, AISec'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 111. doi:10.1145/3338501.3357370.
URL https://doi.org/10.1145/3338501.3357370

[117] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, S. Liu, Efficient and privacy-enhanced federated learning for industrial artificial intelligence, IEEE Transactions on Industrial Informatics 16 (10) (2019) 6532–6542.

[118] M. Hao, H. Li, G. Xu, S. Liu, H. Yang, Towards efficient and privacy-preserving federated deep learning, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–6. doi:10.1109/ICC.2019.8761267.

[119] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 3742. doi:10.1145/2757384.2757397.
URL https://doi.org/10.1145/2757384.2757397

[120] S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, in: in International Conference on Wireless Algorithms, Systems and Applications (WASA, 2015, pp. 685–695. doi:10.1007/978-3-319-21837-3_67.

[121] S. Wang, J. Xu, N. Zhang, Y. Liu, A survey on service migration in mobile edge computing, IEEE Access 6 (2018) 23511–23528. doi:10.1109/ACCESS.2018.2828102.

[122] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, Q. Li, A survey of virtual machine management in edge computing, Proceedings of the IEEE 107 (8) (2019) 1482–1499. doi:10.1109/JPROC.2019.2927919.

[123] C. Thapa, M. A. P. Chamikara, S. Camtepe, Splitfed: When federated learning meets split learning (2020). arXiv:2004.12088.

[124] P. Vepakomma, O. Gupta, T. Swedish, R. Raskar, Split learning for health: Distributed deep learning without sharing raw patient data (2018). arXiv:1812.00564.

[125] C. Dupont, R. Giaffreda, L. Capra, Edge computing in iot context: Horizontal and vertical linux container migration, in: 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–4. doi:10.1109/GIOTS.2017.8016218.

[126] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, I. Humar, A dynamic service migration mechanism in edge cognitive computing, ACM Trans. Internet Technol. 19 (2). doi:10.1145/3239565.
URL https://doi.org/10.1145/3239565

[127] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, K. Temma, Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration, IEEE Transactions on Computers 67 (9) (2018) 1287–1300. doi:10.1109/TC.2018.2818144.

[128] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–7. doi:10.1109/ICC.2019.8761315.

[129] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, R. Yonetani, Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data, in: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–7. doi:10.1109/ICC40277.2020.9149323.

[130] H. H. Yang, Z. Liu, T. Q. S. Quek, H. V. Poor, Scheduling policies for federated learning in wireless networks, IEEE Transactions on Communications 68 (1) (2020) 317–333. doi:10.1109/TCOMM.2019.2944169.

[131] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, V. Gramoli, Federated learning over wireless networks: Convergence analysis and resource allocation, IEEE/ACM Transactions on Networking (2020) 1–12doi:10.1109/TNET.2020.3035770.

[132] T. Li, M. Sanjabi, A. Beirami, V. Smith, Fair resource allocation in federated learning, in: International Conference on Learning Representations, 2020.
URL https://openreview.net/forum?id=ByexElSYDr

[133] Q. Zeng, Y. Du, K. Huang, K. K. Leung, Energy-efficient radio resource allocation for federated edge learning, in: 2020 IEEE International Conference on Communications Workshops (ICC Workshops), 2020, pp. 1–6. doi:10.1109/ICCWorkshops49005.2020.9145118.

[134] M. J. Neely, E. Modiano, C. Li, Fairness and optimal stochastic control for heterogeneous networks, IEEE/ACM Transactions on Networking 16 (2) (2008) 396–409. doi:10.1109/TNET.2007.900405.

[135] M. S. H. Abad, E. Ozfatura, D. GUndUz, O. Ercetin, Hierarchical federated learning across heterogeneous cellular networks, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 8866–8870. doi:10.1109/ICASSP40776.2020.9054634.

[136] Y. Sun, S. Zhou, D. Gndz, Energy-aware analog aggregation for federated learning with redundant data, in: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–7. doi:10.1109/ICC40277.2020.9148853.

[137] G. Zhu, Y. Wang, K. Huang, Broadband analog aggregation for low-latency federated edge learning, IEEE Transactions on Wireless Communications 19 (1) (2020) 491–506. doi:10.1109/TWC.2019.2946245.

[138] Y. Zou, S. Feng, D. Niyato, Y. Jiao, S. Gong, W. Cheng, Mobile device training strategies in federated learning: An evolutionary game approach, in: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019, pp. 874–879. doi:10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00157.

[139] H. T. Nguyen, N. C. Luong, J. Zhao, C. Yuen, D. Niyato, Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach, CoRR abs/1910.09172. arXiv:1910.09172.
URL http://arxiv.org/abs/1910.09172

[140] Y. Zhan, P. Li, S. Guo, Experience-driven computational resource allocation of federated learning by deep reinforcement learning, in: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2020, pp. 234–243. doi:10.1109/IPDPS47924.2020.00033.

[141] Y. Chen, Y. Ning, H. Rangwala, Asynchronous online federated learning for edge devices, CoRR abs/1911.02134. arXiv:1911.02134.
URL http://arxiv.org/abs/1911.02134

[142] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Differentially private asynchronous federated learning for mobile edge computing in urban informatics, IEEE Transactions on Industrial Informatics 16 (3) (2020) 2134–2143. doi:10.1109/TII.2019.2942179.

[143] Y. Chen, X. Sun, Y. Jin, Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation, IEEE Transactions on Neural Networks and Learning Systems 31 (10) (2020) 4229–4238. doi:10.1109/TNNLS.2019.2953131.

[144] T. Chen, X. Jin, Y. Sun, W. Yin, Vafl: a method of vertical asynchronous federated learning (2020). arXiv:2007.06081.

[145] S. Feng, D. Niyato, P. Wang, D. I. Kim, Y. Liang, Joint service pricing and cooperative relay communication for federated learning, in: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019, pp. 815–820. doi:10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00148.

[146] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, C. S. Hong, Federated learning for edge networks: Resource optimization and incentive mechanism, IEEE Communications Magazine 58 (10) (2020) 88–93. doi:10.1109/MCOM.001.1900649.

**Declaration of interests**

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: