Knowledge and Information Systems (2022) 64:885–917
https://doi.org/10.1007/s10115-022-01664-x

SURVEY PAPER

From distributed machine learning to federated learning: a survey

Ji Liu[1] · Jizhou Huang[1] · Yang Zhou[2] · Xuhong Li[1] · Shilei Ji[1] · Haoyi Xiong[1] · Dejing Dou[1,3]

Received: 18 March 2021 / Revised: 31 January 2022 / Accepted: 5 February 2022 /
Published online: 22 March 2022

Abstract
In recent years, data and computing resources are typically distributed in the devices of end users, various regions or organizations. Because of laws or regulations, the distributed data and computing resources cannot be aggregated or directly shared among different regions or organizations for machine learning tasks. Federated learning emerges as an efficient approach to exploit distributed data and computing resources, so as to collaboratively train machine learning models. At the same time, federated learning obeys the laws and regulations and ensures data security and data privacy. In this paper, we provide a comprehensive survey of existing works for federated learning. First, we propose a functional architecture of federated learning systems and a taxonomy of related techniques. Second, we explain the federated learning systems from four aspects: diverse types of parallelism, aggregation algorithms, data communication, and the security of federated learning systems. Third, we present four widely used federated systems based on the functional architecture. Finally, we summarize the limitations and propose future research directions.

Keywords Federated learning · Distributed system · Parallel computing · Security · Privacy

1 Introduction

With billions of connected Internet-of-Things (IoT) devices [3], smartphones [132] and large websites around the world, in recent years, we have witnessed huge amounts of data generated and dispersed over various mobile devices of end users, or the data centers of different organizations. As the data contain sensitive information of end users or organizations, such

✉ Ji Liu
liuji04@baidu.com

✉ Dejing Dou
doudejing@baidu.com

1 Baidu Inc., Beijing, China

2 Computer Science and Software Engineering Department, Auburn University, Auburn, AL, USA

3 Computer and Information Science Department, University of Oregon, Eugene, OR, USA

as facial images, location-based services, health information [114], or personal economic status, moving the raw data from personal devices or data centers of multiple organizations to a centralized server or data center may pose immediate or potential information leakage. Due to the concerns of data security and data privacy, legal restrictions, such as the Cybersecurity Law of the People's Republic (CLPR) of China [160], the General Data Protection Regulation (GDPR) [133] in European Union, the Personal Data Protection Act (PDP) [31] in Singapore, the California Consumer Privacy Act (CCPA) [20], and the Consumer Privacy Bill of Rights (CPBR) [47] in the USA, have been introduced and put in practice, which makes data aggregation from distributed devices, multiple regions, or organizations, almost impossible [190]. In addition, computing and storage resources are also typically distributed in multiple regions [106] and organizations [98], which cannot be aggregated in a single data center.

Federated learning (FL) emerges as an efficient approach to exploit the distributed resources to collaboratively train a machine learning model. FL is a distributed machine learning approach where multiple users collaboratively train a model, while keeping the raw data decentralized without being moved to a single server or data center [79,190]. FL not only exploits the distributed resources to efficiently carry out the training process of machine learning, but also promises to provide security and privacy for the decentralized raw data. Within FL, the raw data, or the data generated based on the raw data with security processing, serves as the training data. FL only allows the intermediate data to be transferred among the distributed computing resources while avoiding the transfer of training data. The distributed computing resources refer to mobile devices of end users or servers of multiple organizations. FL brings the code to the data, instead of bringing the data to the code, and it addresses the fundamental problems of privacy, ownership, and locality of data [120]. In this way, FL can enable multiple users to train a model while satisfying the legal data restrictions.

Traditional centralized machine learning approaches typically gather the distributed raw data generated on different devices or organizations to a single server or a cluster with shared data storage, which may bring serious data privacy and security concerns [205]. The centralized approaches, in general, are associated with diverse challenges, including computational power and training time, and most importantly, security and privacy with respect to distributed data [129]. FL differs from the centralized approach in three aspects. First, FL does not allow direct raw data communication, while the centralized approach has no restriction. Second, FL exploits the distributed computing resources in multiple regions or organizations, while the centralized approach generally only utilizes a single server or a cluster in a single region, which belongs to a single organization. Third, FL generally takes advantage of encryption or other defense techniques to ensure the data privacy or security, while the centralized approach pays little attention to these security issues [205].

The term "federated learning" was first introduced in 2016 [120], which focuses on the unbalanced and non-Independent and Identically Distributed (non-IID) data in mobile devices. The concept of FL was extended to three data scenarios, i.e., horizontal, vertical, and hybrid [190,205]. The horizontal FL addresses the decentralized data of the same features, while the identifications are different. The vertical FL handles the decentralized data of the same identifications with different features. The hybrid FL deals with the data of different identifications and different features. Then, FL is formally defined as a machine learning approach where multiple clients collaborate in solving a machine learning problem while the raw data is stored locally and is neither exchanged nor transferred [79].

An FL system is an efficient tool to carry out FL with decentralized data and resources. Several open-source FL systems, e.g., FATE [182], PaddleFL [10], TensorflowFL [55], and Pysyft [136], are now intensively used by both research communities, e.g., healthcare [19],

and computer visions [66,109], and by industrial groups, e.g., WeBank [183]. Although various FL systems exist, the architecture of FL systems has common features: In particular, they share the capability to collaboratively train a machine learning model. Most FL systems are composed of four layers, i.e., presentation, user services, FL training, and infrastructure. These four layers enable FL system users to design, execute, and analyze machine learning models with distributed data.

Although FL differs from the centralized machine learning approaches, it not only utilizes novel techniques designed for FL, but also takes advantage of the techniques designed for distributed machine learning. FL exploits parallelization techniques designed for distributed machine learning. For instance, horizontal FL exploits the data parallelism, which trains multiple instances of the same model on different subsets of the training dataset [170]. Vertical FL utilizes model parallelism to distribute parallel paths of a single model to multiple devices in order to handle the data of different features [170]. Multiple aggregation algorithms [29] are proposed to aggregate the models in distributed computing resources. Data transfer techniques are also utilized in FL, e.g., model compression [22]. As FL promises to provide data security and data privacy, diverse defense techniques, e.g., differential privacy [121], homomorphic encryption [59], and robustness aggregation [142], are designed to address the possible attacks [49,71,181].

There have been a few surveys of FL. Some works [79,89,190] provide a comprehensive study of FL, from the taxonomy of FL to the techniques, e.g., the efficiency, data privacy, security, and applications of FL. Some surveys [89,114,129] focus on the data privacy and security of FL. Other surveys present the application of FL in a specific area, e.g., healthcare informatics [188], mobile edge networks [99], and neural architecture searches [205], and they personalize global models to work better for individual clients [87]. However, few of them present the architecture of FL or analyze parallelization techniques in FL.

In this paper, we provide a survey of federated learning and the related parallelization techniques. The main contributions of this paper are:

- A four-layer FL system architecture, which is useful for discussing the techniques for FL. This architecture can also be a baseline for other work and can help with the assessment and comparison of FL systems.
- A taxonomy of FL-related techniques, including the parallelization techniques, the aggregation algorithms, and the techniques for data communication and security, with a comparative analysis of the existing solutions.
- A discussion of research issues to improve the efficiency and security of FL systems.

This paper is organized as follows: Section 2 gives an overview of the execution of FL, including the FL system architectures and basic functional architecture of FL systems. Section 3 focuses on the techniques used for distributed training of FL and aggregation methods. Section 4 presents the techniques for distributed execution, data communication, and data security of FL. Section 5 demonstrates the existing FL frameworks. Section 6 discusses the open issues raised for the execution of FL with distributed resources. Section 7 summarizes the main findings of this study.

2 An overview of federated learning

In this section, we introduce the basic concepts of federated learning. Then, we present the life cycle of FL models. Afterward, we detail the functional architecture and the corresponding functionality of FL systems.

2.1 Basic concepts

Machine learning is the process to automatically extract the models or patterns from data [53]. The models or patterns are expressed as machine learning models. A machine learning model is an ensemble of a model structure, which is typically expressed as a directed acyclic graph (DAG), data processing units, e.g., activation functions in deep neural networks (DNNs), and the associated parameters or hyper-parameters. The input data can be processed through a machine learning model to generate the output, e.g., the prediction results or the classification results, which is the inference process. The machine learning model is generated based on the training data, which is the training process. During the training process, the parameters or the model structure of the machine learning model [61,68] is adjusted based on a training algorithm in order to improve the performance, e.g., the accuracy or the generalization capacity. The training algorithm is also denoted by machine learning algorithms. The duration of the training process is training time.

According to whether the training data have labels, the training process of machine learning can be classified into four types [170], i.e., supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning represents that a machine learning task exploits the training data composed of input features and the corresponding labels [170]. In this paper, we focus on this type of training data. For instance, each data point in the training dataset contains (x, y), where x represents the input features and y represents the desired output value. Unsupervised learning represents that a machine learning task exploits the training data, which only consists of input features without output values, i.e., each data point only contains x and does not have y. Semi-supervised learning represents that one (generally small) part of the training data contains output values, while the other (generally small) part of the training does not. Reinforcement learning represents that each iteration in the training process considers its observation of the environment from the last iteration.

While the training data become huge, e.g., on the order of terabyte [23], or when the training data are inherently distributed or too big to store on single machines [170], the training process is carried out using distributed resources, which is distributed machine learning. One of the important features of the distributed machine learning is that it can significantly accelerate the training speed so as to reduce the training time. Diverse parallelization techniques are used in distributed machine learning. For instance, graphics processing units (GPUs) using single instruction multiple data (SIMD) [45] and tensor processing units (TPUs) using multiple instructions multiple data (MIMD) [45] are exploited [170]. In addition, distributed machine learning takes advantage of three types of parallelism to parallelize the training process, i.e., data parallelism [170], model parallelism [170], and pipeline parallelism [64,74,105]. With the data parallelism approach, the training data are partitioned as many times as the number of computing resources, and all computing resources subsequently apply the same machine learning algorithm to process different chunks of the data sets [170]. With the model parallelism approach, exact copies of the entirety of the data (the training data or the intermediate data) are processed by each computing resource, each of which exploits different parts of the machine learning model [170]. The pipeline parallelism approach combines the data parallelism and the model parallelism. With this approach, each computing resource processes a part of the training data with a part of the machine learning model, while the processing, e.g., computation or communication, at each node can be parallelized [131].

FL is a distributed machine learning approach where multiple users collaboratively train a model, while keeping the raw data distributed without being moved to a single server or data center [79,190]. The model used for FL is denoted by FL model. FL is first proposed to

handle the unbalanced and non-Independent and Identically Distributed (non-IID) data of the same features in mobile devices [120]. Then, the concept of FL is extended to the distributed data of diverse features in multiple organizations [190] or various regions [80]. FL systems are used within one or multiple phases of the life cycle of FL models. An FL system is a distributed system to manage the distributed training process with distributed resources.

FL is a special type of distributed machine learning, which differs from other distributed machine learning approaches in the following three points. First, FL does not allow direct raw data communication, while other approaches have no restriction. As the raw data are of multiple ownerships, FL approaches with this restriction can meet the requirements defined by the related laws, e.g., CLPR [160], GDPR [133], PDPA [31], CCPA [20], and CPBR [47]. In particular, the consent (GDPR Article 6) and the data minimalization principle (GDPR Article 5) limit data collection and storage to only what is consumer-consented and what is absolutely necessary for processing [99]. Second, FL exploits the distributed computing resources in multiple regions or organizations, while the other approaches generally only utilize a single server or a cluster in a single region, which belongs to a single organization. FL enables the collaboration among multiple organizations. Third, FL generally takes advantage of encryption or other defense techniques to ensure the data privacy or security, while the other approaches pay little attention to this security issue [205]. FL promises to ensure the privacy and security of the raw data, as the leakage of information may incur significant financial [34,150] and reputational [158] losses.

During the training process of FL, an optimization problem is solved as shown in Formula (1). Given n training dataset $\mathcal{D} = D_1, D_2, ..., D_n$, where each data point $(x, y) \sim \mathcal{D}$, the problem of FL is to learn a function $\widehat{F}$ from all possible hypotheses $\mathcal{H}$, while minimizing the expectation of loss over the distribution of all the dataset $\mathcal{D}$.

$$\widehat{F} = \underset{F \in \mathcal{H}}{\operatorname{argmin}} \underset{(x,y)\in\mathcal{D}}{\mathbb{E}} L(y, F(x)), \tag{1}$$

where $L(y, F(x))$ refers to the loss of $F(x)$ to the label y. During the training process, the stochastic gradient descent (SGD) approach [145,206] is generally used to minimize the loss function using Formula (2).

$$F_{k+1}(x) \leftarrow F_k(x) - \eta_k \nabla F_k(x), \tag{2}$$

where $F_k(x)$ refers to the learned model in the k^{th} iteration, $\nabla F_k(x)$ refers to the gradient of the model at the k^{th} iteration based on the model already obtained $F_k(x)$ and the training dataset, η_k refers to the learning rate, and $F_{k+1}(x)$ refers to the update model of the k^{th} iteration. Within each iteration, there are two phases, i.e., forward propagation and backward propagation. The forward propagation calculates the output based on the input data x using the model, while the backward propagation calculates the gradients $\nabla F_k(x)$ and updates the model. When the calculation is distributed among multiple computing resources, the gradients or models of each computing resource are aggregated using an aggregation algorithm (see details in Sect. 3.2), in order to achieve consensus of multiple models and to generate a global model. The learning rate can be dynamically adapted using a local adaptive optimizer, e.g., Adam, and/or cross-round learning rate schedulers [199].

2.2 FL model life cycle

The life cycle of an FL model is a description of the state transitions of an FL model from creation to completion [79,105]. Lo et al. [111] propose that the life cycle of an FL model consists of eight phases: initiated, broadcast, trained, transmitted, aggregated, evaluated,

deployed, and monitored. Kairouz et al. [79] propose that the life cycle of an FL model includes six phases: problem identification, client instrumentation, simulation prototyping, federated model training, model evaluation, and deployment. However, they focus on the FL with distributed data in mobile devices. In this paper, we adopt a combination of workflow life cycle views [79,111] with a few variations [105,190], condensed into four phases:

1. The composition phase [79] is for the creation of an FL model, which is used to address a specific machine learning problem, e.g., classification. First, a machine learning model is created to address the problem with certain requirements, e.g., the requirement of accuracy. Then, the machine learning model is adapted to FL scenarios. For instance, if the distributed data is of different features, the machine learning model is partitioned (see details in Sect. 3.1.2) to process the distributed data.
2. The FL training phase [79,111,190] is for the training phase of the FL model. During this phase, a training strategy, which includes parallelism and aggregation algorithms (see details in Sect. 3), is used to update the parameters, hyper-parameters, and even the structure of the network, in order to improve the accuracy and the generalization capacity of the FL model.
3. The FL model evaluation phase [79,105] is to apply the trained FL models, in order to analyze the performance of the trained FL models on a simulation platform or a real distributed system [65]. As a result, the FL models with the best performance are selected. If the FL models do not meet the requirements, the FL model should be modified, or the training phase should be carried out again.
4. The FL model deployment phase [79] is to deploy the FL model in a real-life scenario to process the data. If the final model can be shared without restriction, there is no difference between the FL model deployment and the model generated from a traditional centralized approach. Otherwise, the deployment of the final model should consider the ownership of the corresponding parts.

2.3 Functional architecture of FL systems

The functional architecture of an FL system can be layered as follows [105]: presentation, user services, FL training, and infrastructure. Figure 1 shows this architecture. The higher layers exploit the lower layers to provide their own functionality. A user interacts with an FL system through the presentation layer and realizes independent functionalities at the user services layer. During the training phase of FL models, a federated learning execution plan (FLEP) is generated, and the corresponding distributed training is carried out at the FL training layer. The FLEP is composed of a type of parallelism, a scheduling strategy, and a fault-tolerance mechanism. The FL system manages the physical resources through the infrastructure layer for the distributed training.

2.3.1 Presentation layer

The presentation layer is a user interface (UI) for the interaction between users and FL systems at one or multiple stages of the FL model life cycle. The UI can be textual or graphical. This interface is responsible for designing a new FL model or choosing an existing machine learning model as an FL model. In addition, this layer also supports the modules at the user services layer, e.g., shows the status of the distributed training process. The textual UI is largely used for designing FL models based on the command line or scripts [100]. The models can be directly expressed using Python, with the textual interface in PaddleFL [10],

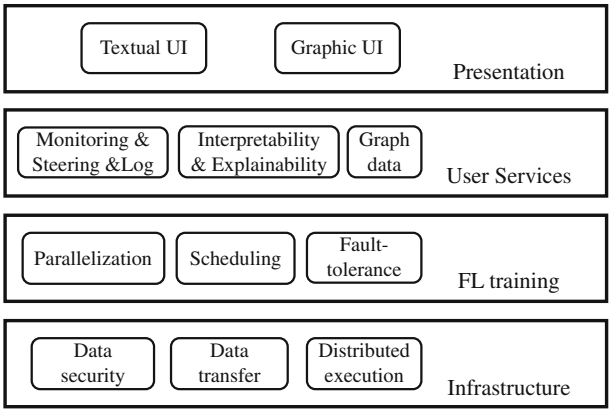


Fig. 1 Functional architecture of an FL system

TensorFlowFL [55], PySyft [148], and FATE [182]. A graphic UI can make the interaction more practical, while the users can drag or drop the data processing element to design an FL model. For instance, FATE [182] provides a graphic UI (GUI) through a web portal. However, the graphic portal also exploits textual programming languages as inner representations of an FL model.

2.3.2 User services layer

The user service layer supports the expected functionalities, i.e., monitoring and steering and log; interpretability and explainability; and graph data. The monitoring enables the users to get the real-time status of the distributed training process. As the training process of FL models can be very long, e.g., from several hours to days [79], it is of much importance to track the execution status, which allows the user to verify whether the training proceeds normally. The log service is generally supported by major FL systems, which can be used to analyze the training process. In addition, the log generated during the training process can be used to debug the system or adjust the FL model. FATE provides a visual monitoring board to users through its GUI. When there are unexpected results or errors during the training process, steering enables users to adjust the training process in order to reduce the time necessary to carry out the distributed training from scratch. Most FL systems can enable the users to stop the training, while the adjustment of parameters is not fully supported by major FL systems. The interpretability of FL is to describe the internals of an FL system in a way that is understandable to humans [52]. The explainability focuses on explaining the representation of data inside an FL model [52]. With interpretability and explainability, the FL system can provide a description of the results of the trained FL model based on the training data and the distributed training process. Shapley values have been used to provide the interpretability [175], while both the interpretability and explainability remain open challenges as each is hard to fully support.

In the real world, graph data widely exist in multiple domains, and a bunch of FL approaches have been proposed to handle the decentralized graph data for community detection [84], financial crime [164], and especially knowledge graph completion [28]. FL is particularly useful in the field of knowledge graph completion, as a knowledge graph could not only contain text but also images or other type of data, i.e., multimodal knowledge graphs

[202], and the completion is realized in a collaborative fashion within an FL system [102]. The decentralized graphs can be inter-graph, i.e., the decentralized data belong to multiple graphs, or intra-graph, i.e., the decentralized data are within one big graph [198], while most of the existing works focus on the intra-graph situation. Horizontal FL techniques can be exploited on the graph neural networks (GNNs) [125,185] with encryption techniques [78] (see details in Sect. 2.3.4), while the performance of FL may be much worse than that of centralized GNNs [62]. The aggregation algorithms (see details in Section 3.2.1) are also proposed based on the FedAvg [202] or optimal transportation [102]. In addition, decentralized aggregation algorithms (see details in Sect. 3.2.3) are also proposed to deal with the decentralized graph data for social network [67] and drug discovery [63]. While the fine-tuning of the FL system is time-consuming, Bayesian optimization [203] and evolutionary optimization strategies [174] are utilized to automatically tune the hyper-parameters and the network structure, respectively. Graph data can be vertically distributed where the features of the nodes are distributed across multiple data owners, and a data owner may or may not have the edges. Vertical FL exploits embeddings [28,204] or autoencoders [202] to represent the nodes, which can be transferred to train a GNN. In addition, the differential privacy (see details in Sect. 2.3.4) is combined with the embeddings to protect the data privacy [140] of knowledge graphs.

2.3.3 FL training layer

The FL training layer carries out the distributed training process with distributed data and computing resources. This layer consists of three modules: parallelization, scheduling, and fault-tolerance. FL parallelization exploits diverse types of parallelism, e.g., data parallelism, model parallelism, and pipeline parallelism, to generate executable tasks. Through the FL scheduling module, an FL system produces a scheduling plan (SP) of executable tasks, which aims at fully exploiting distributed computing resources and preventing training stalling. During the training process, the SP is generally defined by a training algorithm, which aggregates the updates, i.e., gradients or models, from each computing resource in order to generate a final machine learning model. The FL fault-tolerance mechanism handles the failures or errors of task execution and the connection of distributed resources. Reactive approaches are generally exploited, e.g., using check-points, restart, and task replication [17]. A reactive approach reduces the effect of failures after perceiving failures [48]. An FLEP, which captures the execution directives, typically the result of compiling and optimizing the training process of FL models, is generated at this layer.

2.3.4 Infrastructure layer

The infrastructure layer provides the interaction between an FL system and the distributed resources, including the computing resources, storage resources, network resources, and data resources. This layer contains three modules: a data security module, a data transfer module, and a distributed execution module. The data security module generally exploits differential privacy (DP) [2] and encryption techniques, e.g., homomorphic [4], to protect the raw data used during the training process. Although the raw data cannot be directly transferred, intermediate data, e.g., the gradients or models, can be communicated among distributed computing resources. The data transfer module exploits data compression techniques [161] to improve the data transfer efficiency. At this layer, the FLEP generated at the FL training layer is carried out within the distributed execution module, i.e., concrete tasks are executed in distributed computing resources.

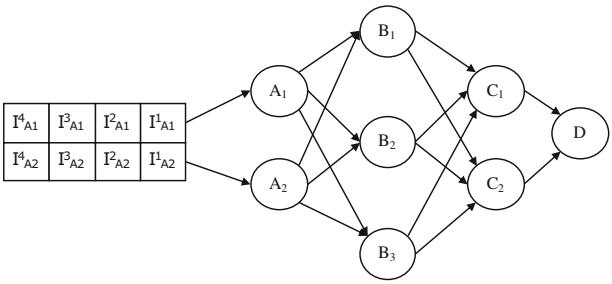


Fig. 2 An example of a neural network

3 Distributed training

In this section, we present the distributed training process for FL. First, we present three types of parallelism in distributed training and the application within FL. The parallelism approaches are generally implemented in the parallelization module. Then, we discuss existing aggregation algorithms for the distributed training, which is implemented in the scheduling module.

3.1 Parallelism & FL types

Three types of parallelism exist for distributed machine learning: data parallelism, model parallelism, and pipeline parallelism [105,170]. FL can be classified to three types, i.e., horizontal, vertical, and hybrid [190,205]. The horizontal FL generally exploits data parallelism, and the vertical FL typically takes advantage of model parallelism. However, the hybrid FL relies on transfer learning [139], which is not a parallelism approach and is out of the scope of this paper.

In this section, we take an example of a neural network as shown in Fig. 2 to explain the parallelism. In the example, we assume that the model contains three layers and seven data processing nodes (neurons), i.e., A_1, A_2, B_1, B_2, B_3, C_1, C_2, D. The arrows represent the data flow among different data processing nodes. The execution of the data processing nodes at each layer can be carried out in parallel, while the execution of different layers should be performed sequentially. The input data contain 4 data points. We assume two/three computing resources owned by two/three users. Each has a part of the input data.

3.1.1 Data parallelism

Data parallelism is realized by having the data processing performed in parallel at different computing resources, with the same model, on different data points. As shown in Fig. 3, data parallelism is exploited when the ensemble of data points is distributed among different computing resources. During the training process of FL, the training data is not transferred among different computing resources, while the intermediate data, e.g., the models or the gradients $\nabla F_k(x)$ in Formula (2), are transferred. The data in each computing resource can be Independent and Identically Distributed Data (IID) or non-IID. FL focuses on the non-IID [120], while other distributed machine learning approaches mainly focus on IID data. With the data parallelism, the FL is horizontal [190], i.e., the data and the calculation are horizontally distributed among multiple computing resources. In addition, this parallelism

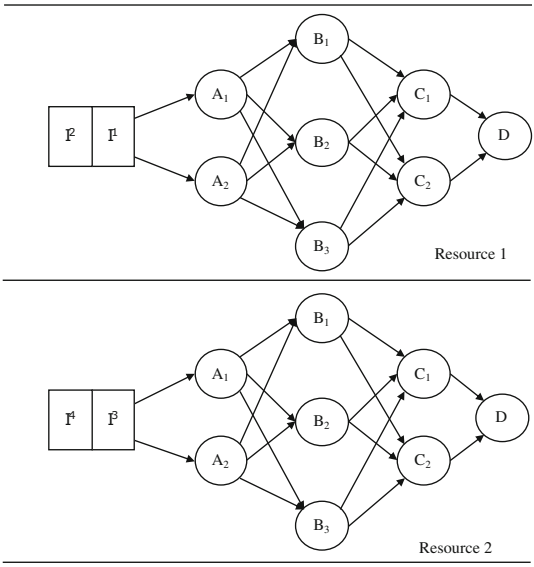


Fig. 3 Data parallelism. The forward and backward process of I^1 and I^2 is performed in computing resource 1, while that of I^3 and I^4 is performed in computing resource 2 at the same time. Then, the model or gradient is transferred to calculate an average model or gradient to be sent to each computing resource for the following training

generally corresponds to the cross-device FL [79], where a large number of devices (mobiles or edge devices) collaboratively participate in training a single global model to have good accuracy. When the number of devices is small, e.g., 2–100, and the computing resources are from diverse organizations, this parallelism also corresponds to cross-silo FL [79]. In addition to the general data-parallel schemes for federated learning, some specific privacy-preserved distributed statistical tricks have been invented for federated sparse models [15,16].

3.1.2 Model parallelism

Model parallelism is realized by having independent data processing nodes distributed at different computing resources, so as to process the data points of specific features. Two data processing nodes can be either independent, i.e., the execution of any node does not depend on the output of the other one, or dependent, i.e., there is a data dependency between them [105]. As shown in Fig. 4, model parallelism is achieved when different parts of each data point are distributed at different computing resources. For instance, the data process on Node A_1 and that of A_2 can be carried out in parallel. With the model parallelism, vertical FL, where the data points and calculation are vertically distributed among multiple computing resources [59,190], is realized. In this case, the original model needs to be partitioned to be distributed at different computing resources. Two organizations generally apply this type of FL when each organization owns parts of the features of users and they would like to collaboratively train a model using the data of all the features, which corresponds to cross-silo FL [79]. Most studies of vertical federated learning only support two parties (with or without a central coordinator) [205]. For instance, SecureGBM [43] is proposed to train a tree-based gradient boosting machine (GBM). In order to support multiple parties, the idea

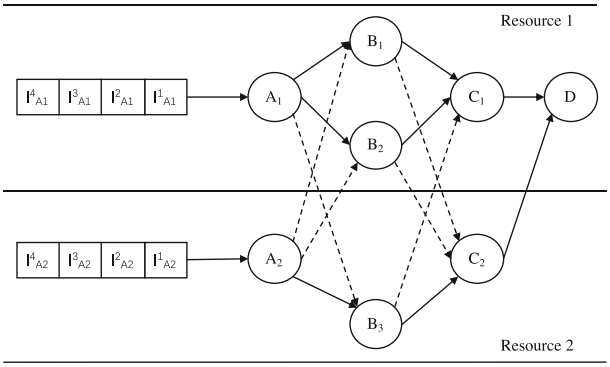


Fig. 4 Model parallelism. The dashed arrows represent inter-computing resource communication. For each input data point I, different parts, i.e., I_{A_1} and I_{A_2}, are distributed at different computing resources

of multi-view learning [187] is exploited in a multi-participant, multi-class vertical federated learning framework [42].

3.1.3 Pipeline parallelism

Pipeline parallelism is realized by having dependent data processing nodes distributed at different computing resources [74,131]. As shown in Fig. 5, the data processing nodes are distributed at multiple computing resources. While data point I_A^3 is processed in computing resource 1, the outputs of A_1 and A_2 are processed in computing resource 2, and the outputs of B_1, B_2, and B_3 are processed in computing resource 3. With this type of parallelism, the dependent data processing nodes can process the data in parallel. As this parallelism may incur many inter-computing resource data transfers, it is not widely used for FL.

3.2 Aggregation algorithms

With the horizontal FL and data parallelism, aggregation algorithms are used to aggregate the models or gradients generated from the forward and backward propagation in each computing resource. The aggregation algorithms can be either centralized, or hierarchical, and decentralized. The centralized aggregation algorithms generally rely on a centralized server, i.e., a parameter server, to synchronize or schedule the execution of distributed computing resources, while hierarchical aggregation algorithms rely on multiple parameter servers for the model aggregation. The decentralized aggregation algorithms make each computing resource equally perform the calculation based on a predefined protocol, without relying on a centralized server. Please refer to [177] for the details of federated optimization. The characteristics are summarized in Table 1, which can be used to choose appropriate algorithms in a specific situation.

3.2.1 Centralized aggregation

As shown in Fig. 6, a single parameter server is used to calculate the average models or gradients sent from multiple computing resources (mobiles). The weights of the model (model) or the gradients are calculated in each computing resource, which are transferred

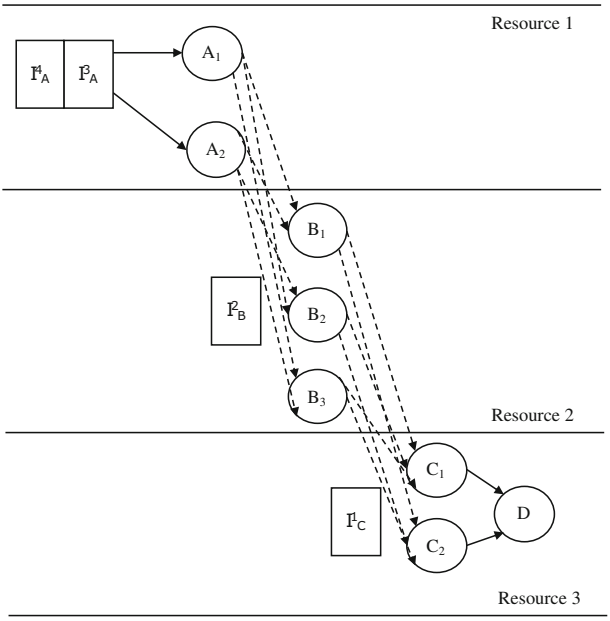


Fig. 5 Pipeline parallelism. The dashed arrows represent inter-computing resource communication

Table 1 Comparison among diverse types of aggregation algorithms. "Complexity" represents the complexity to implement the algorithms ("H" represents high complexity, "M" represents medium complexity, and "L" represents low complexity)

Type	Complexity	Trust	Imbalance	High latency
Centralized	L	Y	N	N
Hierarchical	M	Y	Y	Y
Decentralized	H	N	N	Y

Trust represents whether the aggregation algorithms require that the data owners trust the centralized server, *Imbalance* represents whether the algorithms can address the unbalanced data, *High latency* represents whether the algorithms can support the high-latency model or gradient data transfer, *Y* represents that the algorithms support the functionality, while *N* represents that the algorithms do not support the functionality

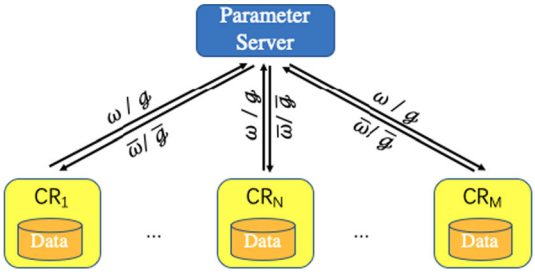


Fig. 6 The architecture of centralized aggregation. "CR" represents computer resource. ω represents the local parameters or the weights of the model calculated in each computing resource. g represents the local gradients in backward propagation in each computing resource. $\overline{\omega}$ represents the global model calculated in the parameter server. $\overline{g}$ represents the global gradients calculated in the parameter server

to a parameter server. The parameter server calculates global gradients or global models according to a centralized aggregation algorithm. The global gradients or global models are transferred to each computing resource for the following computation. The update of the model is based on the SGD defined in Formula (2) in both computing resources, or on the parameter server.

A bunch of centralized aggregation algorithms have been proposed. Federated Averaging (FedAvg) [120] algorithm is introduced as the aggregation method in Google's implementation of an FL system. A centralized server aggregates the machine learning models from selected users. Then, a global model is generated using a weighted sum of each aggregated machine learning model. Afterward, the global model is shared with selected users, and the training process is continued in the computing resource of selected users. However, the trained model of FedAvg may be biased toward computing resources with favorable network conditions [91]. While FedAvg is a straightforward approach, some other methods are proposed to address additional problems. A Federated Stochastic Block Coordinate Descent (FedBCD) [110] algorithm is proposed to reduce the number of communication rounds by enabling multiple local updates before the model communication between a user and the server. In addition, FedBCD also considers the regularization during the training process. The training problem with regularization can be formulated as:

$$\widehat{F} = \underset{F_\theta \in \mathcal{H}}{\operatorname{argmin}} \underset{(x,y)\sim\mathcal{D}}{\mathbb{E}} L(y, F_\theta(x)) + \lambda \cdot \gamma(\theta), \tag{3}$$

where F, D, $\mathcal{H}$ are the same as those in Formula (1), while $\gamma(\cdot)$ denotes the regularizer and λ is the hyper-parameter. The regularization is exploited to improve the generalization capacity of the trained machine learning model. As the fairness among multiple users is important for an FL system, the Stochastic Agnostic Federated Learning (SAFL) [128] algorithm and the FedMGDA+ [73] algorithm are proposed to achieve fairness during the training process of FL. The fairness represents that the data distribution among multiple users can be equally considered without the influence of unrelated factors. Fairness may also refer to two other concepts: (1) a user gets a final model according to the contribution [115] and/or (2) uniform accuracy distribution among all the distributed computing resources [93], which are out of the scope of this paper. In addition, while the computing resources may be heterogeneous, FedProx [92] is proposed to tackle the heterogeneity in an FL system. FedProx enables multiple iterations in each computing resource, while minimizing a cost function based on the local loss function and the global model. Furthermore, in order to address permutation of data processing nodes during the training process, Federated Matched Averaging (FedMA) [176] is proposed. FedMA exploits an existing approach, i.e., BBP-MAP [193], to generate a matrix, in order to align the data processing nodes of the models from computing resources and the server. SCAFFOLD [81] is proposed to reduce the communication rounds, using stateful variables in the distributed computing resources. Attention-augmented mechanism is exploited in Attentive Federated Aggregation (FedAttOpt) [77] to aggregate the knowledge generated from each computing resource (client), based on the contribution of the model from each client. When the data distribution is heterogeneous among users, personalization remains an open problem. In order to address this problem, the model can be split into local layers and global layers, which has been proposed in adaptive personalized federated learning (APFL) [36], FedPer [5], and pFedMe [37]. The local layers are trained with the decentralized data in each computing resource of users, while the global layers are trained in the computing resources of users and the server. However, it is difficult to choose a dataset and its partition among clients to measure the personalization brought by APFL or FedPer, so as to prove the improvement compared with FedAvg. The attention-augmented mechanism helps reduce the

Table 2 Comparison among aggregation algorithms

Algorithm	Reg	Fairness	Heterogeneity	Permutation	C-E
FedAvg	N	N	N	N	N
FedBCD	S	N	N	N	S
SAFL	N	S	N	N	N
FedMGDA+	N	S	N	N	S
FedProx	N	N	S	N	S
FedMA	N	N	N	S	S
SCAFFOLD	N	N	N	N	S
FedAttOpt	N	N	N	N	S

Reg represents regularization, *Heterogeneity* represents that the computing resources are heterogeneous, *Fairness* represents that the data distribution among multiple users can be equally considered without the influence of unrelated factors, *Permutation* refers to the permutation of data processing nodes during the training process, *C-E* represents communication efficient, *S* represents that the algorithm supports the functionality, while *N* represents that the algorithm does not have support

communication rounds. In addition, knowledge distillation can also be exploited to aggregate the models, while requiring that there is data in the centralized server [60]. All these algorithms can handle non-IID data. A comparison among the aforementioned algorithms is proposed in Table 2.

3.2.2 Hierarchical aggregation

As shown in Fig. 7, a hierarchical architecture is also exploited using multiple parameter servers. A two-layer hierarchical architecture is proposed to reduce the time to transfer models between a parameter server and computing resources [1]. The hierarchical architecture uses a global parameter server (GPS) and multiple region parameter servers. Each region parameter

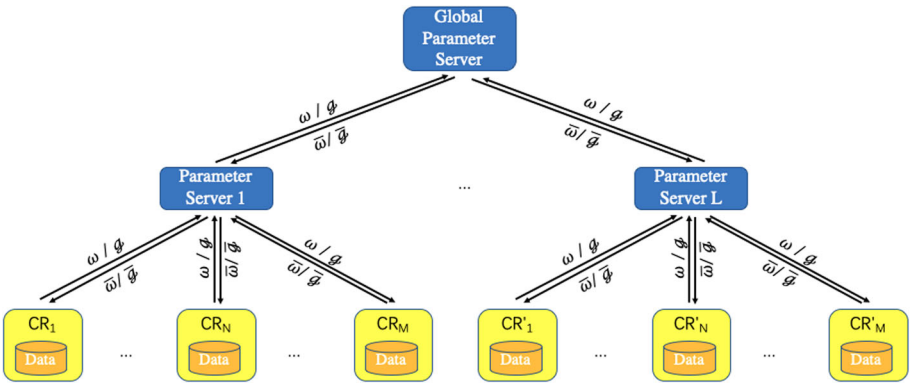


Fig. 7 The architecture of hierarchical aggregation. "CR" represents computer resource. ω represents the local parameters or the weights of the model calculated in each computing resource. g represents the local gradients in backward propagation in each computing resource. $\overline{\omega}$ represents the region or global model calculated in each parameter server. $\overline{g}$ represents the region or global gradients calculated in each parameter server. The region model or gradients are calculated by a region parameter server, while the global model or gradients are calculated by a global parameter server

server (RPS) is implemented in a cell base station where the computing resources (mobiles) can be connected, with low latency. A hierarchical algorithm, i.e., hierarchical federated learning (HFL) is deployed to realize the model aggregation. Within each iteration of HFL, each RPS calculates an average model using the models of the computing resources within its cluster. It sends the averaged model to the GPS, and it receives a global averaged model at every certain iteration. Afterward, it broadcasts the averaged model to all its computing resources. Some other algorithms, e.g., HierFAVG [107], HFEL [112], and LanFL [192], are similar to HFL, while the SPS is an edge or local area network (LAN) parameter server and the MPS is a parameter server implemented on the cloud or a wide area network (WAN). These algorithms take advantage of hierarchical architecture to reduce high-latency model or gradient data transfer, so as to accelerate the training process. In addition, by well-clustering the computing resources to groups, the hierarchical architecture is also exploited to address unbalanced data distributed among multiple computing resources [18,126], or to address data privacy [172].

3.2.3 Decentralized aggregation

While collaboratively training a machine learning model with a decentralized aggregation algorithm, the computing resources can be organized with a *connected* topology and can communicate with a peer-to-peer manner, as shown in Fig. 8. The degree and connectivity of the topology affect the communication efficiency and the convergence rate of the aggregation algorithm. For a given topology, we define $w_{i,j}$, the weight to scale information flowing from node j to node i, as follows

$$w_{ij} \begin{cases} > 0 & \text{if node } j \text{ is connected to node } i, \text{ or } i = j; \\ = 0 & \text{otherwise.} \end{cases} \tag{4}$$

We further define the *topology matrix* $W = [w_{ij}]_{i,j=0}^{n-1} \in \mathbb{R}^{n \times n}$ as the matrix to represent the topology. In the remainder of this paper, we assume that W satisfies $W\mathbb{1} = \mathbb{1}$ and

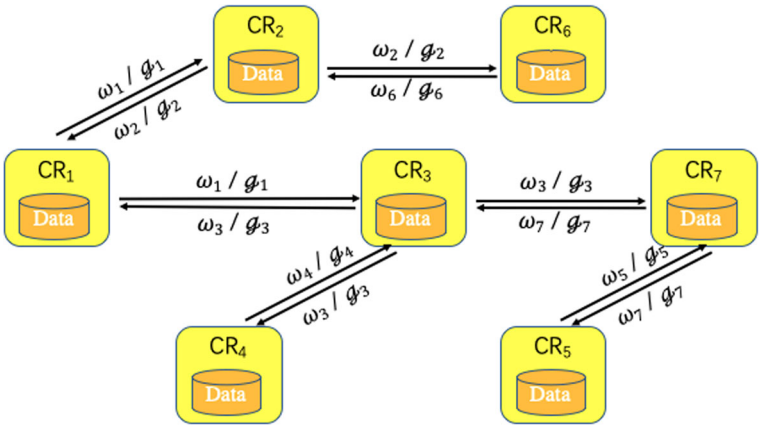


Fig. 8 The architecture of decentralized aggregation. "CR" represents computer resource. ω represents the local parameters, or the weights of the model calculated in each computing resource. g represents the local gradients in backward propagation in each computing resource. When two computing resources are neighbors, they can communicate with each other

$\mathbb{1}^T W = \mathbb{1}^T$, i.e., both the row sum and the column sum of W are equal to 1, so as to guarantee that the neighborhood averaging will asymptotically approach the global averaging [25,151,154]. When a computing resource j is directly connected to computing resource i, i.e., $w_{i,j} \neq 0$, computing resource j is the neighbor of computing resource i. Please note that the weight $w_{i,j}$ denotes the confidence node i has in the information it receives from node j [88], which is different from the bandwidth or data transfer capacity in a network. The centralized aggregation algorithm is a special type of decentralized aggregation with a star topology, while only the centralized server communicates with its neighbors. A well-designed topology, e.g., an exponential graph [6], can improve the convergence rate, which accelerates the training speed.

With the decentralized SGD (D-SGD), each computing resource maintains a local copy of the global model parameters, and it updates the local copy using the models of its neighbors. According to the order to conduct neighborhood averaging and gradient descent, D-SGD has two common types of realizations: average with communication (AWC) [88,96] and average before communication (ABC) [27,178]. AWC can overlap communication and gradient computation, while ABC needs to sequentially calculate and communicate the gradient or model. However, ABC is robust [152], and it converges fast in terms of iterations by exploiting its large learning rate.

In addition, the decentralized aggregation algorithms can be classified to full communication (FC) [96] and partial communication (PC) [168,178] according to the number of neighbors. Within the iterations of FC, each computing resource calculates an averaged model or gradient, based on all the models or gradients of the last version from all its neighbors. However, within the iterations of PC, each computing resource calculates an averaged model or gradient based on one or multiple chosen neighbors. With PC, the selection of the neighbors can be based on a gossip algorithm [72]. For instance, a random neighbor can be selected [168]; the neighbors that provide benign models are selected to avoid attack [130].

4 Data manipulation

At the infrastructure layer of an FL system, there are three types of data manipulation: data security mechanisms, data transfer, and distributed data processing within the distributed execution module. We first present the techniques for the distributed execution in an FL system. Then, we present the techniques for data transfer during the training process of an FL system. Finally, as data security is of much importance to an FL system [129], we present the techniques to protect the data security.

4.1 Distributed data processing

While the bandwidth within a single data center is high, e.g., InfiniBand, the high-performance computing (HPC) libraries, e.g., message passing interface (MPI) [56] or NVIDIA collective communications library (NCCL) [8], are widely exploited for distributed data processing [171]. With MPI or NCCL, the gradients or models in each computing resource can be easily calculated using ring-AllReduce algorithm [51]. However, one of the drawbacks of the HPC libraries is that they lack support for fault-tolerance, as the HPC libraries are designed for high-performance servers with high-quality networks. When any computing resource within the network becomes unavailable, the distributed training process may be broken.

However, as an FL system is generally implemented for the collaboration of large amounts of mobile device users or different organizations, the network connection among computing

resources is of moderate quality, i.e., the bandwidth is not as good as that within a single data center, and the latency is high. For instance, the Internet upload speed is typically much slower than the download speed [86]. Also, some users with unstable wireless communication channels may consequently drop out due to disconnection from the Internet [99]. In this environment, the connection between computing resources and parameter servers has a high possibility of becoming disabled. Thus, remote procedure call (RPC) frameworks are widely exploited, as this kind of framework can ignore the disconnected computing resources and continue the distributed training of an FL system [12], e.g., PaddleFL [10], PySyft [136], or TensorflowFL [55].

4.2 Data transfer

As the network connection is of moderate quality, the data transfer module mainly focuses on data compression to transfer intermediate data, e.g., gradients or models. Sketched updates are proposed for gradient compression to accelerate the data transfer during the distributed training within a single data center [75,76,82,159]. With the data parallelism and centralized aggregation algorithm, before sending the intermediate data, the intermediate data can be sketched with subsampling [86], quantization [57,86,156,157,162,189], sparsification [117, 162], or projection to lower-dimensional spaces [147], in each computing resource, in order to reduce the cost to transfer data. Subsampling refers to transferring only a random subset of the intermediate data [86]. Quantization methods encode each value using a fixed number of bits, so as to reduce the length of gradients or models [86]. With the sparsification approach, only selected parts of the intermediate data are transferred, while the selection is based on a threshold, e.g., the gradients larger than a threshold are selected [162]. Then, when the intermediate data are received in the server, they are decompressed to be aggregated according to the aggregation algorithms presented in Sect. 3.2.1. The convergence of the quantization approach is analyzed in [57], which shows that this approach can also provide good convergence rates [57]. In addition, irrelevant intermediate data can be precluded to be transferred to the server, in order to substantially reduce the communication overhead [180].

4.3 Data security

Data security is of much importance for data processing. The problem of data security is related to significant financial [34,150] and reputational [158] losses. For instance, Uber had to pay $148,000,000 to settle the investigation incurred by a breach of 600,000 drivers' personal information in 2016 [33]. Data security mainly includes two aspects, i.e., data privacy and model security. Data privacy refers to the protection of raw data to avoid raw data information leakage during or after the distributed training of FL systems. Model security refers to the protection of the security of trained models, in order to avoid wrong output based on the trained models incurred by malicious attacks. In this section, we first present the techniques to protect data privacy. Then, we present the defense methods for model security.

4.3.1 Data privacy

The techniques to protect data privacy consist of three types: trusted execution environment (TEE), encryption, differential privacy (DP), and anti-generative adversarial network (GAN) methods. These techniques can be combined in FL systems, e.g., the combination of DP and

TEE in [58], the combination of encryption and DP [194], and the combination of DP and anti-GAN [166].

A TEE is an environment where the execution is secured and no information can be leaked to unauthorized users. Intel SGX technique [119] has been first proposed as a secure environment while providing a set of security-related instruction codes built within Intel central processing units (CPUs). Then, the implementation of machine learning models has been carried out in the TEE, i.e., Intel SGX, in order to enable collaborative data analysis based on machine learning algorithms while providing a security guarantee [134]. Afterward, the TEE has been exploited in FL systems, in order to protect the privacy of data in two ways. The first way is to put the entire training process in the TEE of each distributed computing resource to protect the data privacy during the distributed training [30,127]. The second way is to use TEE to check a small part of the distributed training, while exploiting insecure computing resources, e.g., GPUs, to reduce the training time [200].

As an encryption technique, homomorphic encryption has been used to ensure the data privacy for FL systems [43,58]. Homomorphic encryption [191] allows specific types of computations to be carried out on encrypted input data and to generate an encrypted result, which matches the result of the same computations on the decrypted input data. Two main branches of homomorphic encryption exist, i.e., fully homomorphic encryption and partially homomorphic encryption. The fully homomorphic encryption supports both addition and multiplication on ciphertext, while partially homomorphic encryption only supports either an addition or a multiplication operation on ciphertext, which corresponds to less computational flexibility and better runtime efficiency. Both the fully and partially homomorphic encryptions can be exploited with the horizontal and vertical federated learning. As sharing gradients also leaks the information of training data in horizontal federated learning [49,95,201], it is of much importance to protect the privacy of the intermediate data. Thus, the intermediate data can be encrypted using a homomorphic encryption algorithm before being sent to a parameter server [115,118]. In this way, the intermediate data remain encrypted during the aggregation process while only the computing resource can decrypt the encrypted intermediate data. Even if the transferred encrypted intermediate data are leaked, the information of gradients or models remains safe, and the privacy of the training data is ensured. In addition, partial homomorphic encryption, e.g., Paillier [138], is exploited in vertical federated learning [24,41]. However, the homomorphic encryption incurs significant costs in computation and communication during distributed training [194]. In order to reduce the overhead of homomorphic encryption, a set of quantized gradients are encrypted [194].

Differential privacy (DP) protects the data privacy by adding artificial noise to a small part of raw data, while ensuring that the modification does not substantially affect the performance of the machine learning models [2,38,50,184]. DP is widely used in FL systems as the first step to process the raw data, and the output is the training data to be used for the distributed training [83,97,108,141,149,184]. With more added noise, the privacy is better protected, i.e., there is less possibility to leak raw data information, while it takes more time to converge for the machine learning models [184]. A trade-off between the privacy protection and the convergence performance can be made by selecting a certain number of distributed resources [153,167,184]. However, DP may not be able to ensure the data privacy under certain attacks, e.g., generative adversarial network (GAN) attacks [71].

A well-trained machine learning model can leak information about the training data based on the intermediate data, e.g., gradients [7,71,124]. GANs can be used to generate data similar to the training data based on a well-trained machine learning model [54] in either a parameter server [181] or a distributed computing resource [71]. The adversary can reconstruct other participating clients' private data, even if it has no knowledge of the label information using

the GANs. Thus, during the distributed training process of FL systems, a malicious user can exploit GANs to infer the training data of other users. DP can be used to prevent the GAN-based attack [71,166]. In addition, fake training data can be generated based on a GAN and original raw data, which is then used during the distributed training process to prevent the GAN-based attack [113].

4.3.2 Model security

We mainly focus on poisoning attacks in this section. The objective of poisoning attacks is to reduce the accuracy of machine learning models using artificially designed data, i.e., data poisoning, or models, i.e., model poisoning, in one or several distributed computing resources during the model aggregation process (see details in Sect. 3.2.1). There are two ways to carry out poisoning attacks, i.e., data poisoning and model poisoning.

Data poisoning can be realized by modifying the features [46] or the labels [165] of the input data. For instance, malicious users can modify the data points of a certain class C to other classes, and they can then use the modified data points to participate in the distributed training. The modification of the labels is denoted by the label flipping attack. As a result, the accuracy of the trained model has low accuracy in terms of Class C [165]. Model poisoning refers to the attacks in which the updated intermediate data, e.g., gradients or models, are poisoned before being sent to a parameter server in order to reduce the accuracy of the trained model [26,163]. The goal of the model poisoning is to reduce the performance of the trained model on targeted tasks or classes, while the performance of the model remains unchanged in terms of other tasks or classes [163]. Data poisoning eventually realizes the model poisoning, as it enables some computing resources to update poisoned intermediate data based on the calculation of poisoned training data [46]. However, model poisoning can be more powerful than data poisoning, as model poisoning directly influences the weights of the models and trains in a way that benefits the attack [9]. Both the data poisoning and the model poisoning rely on the backdoor attacks to modify the training data or the intermediate data [26,46,163]. Backdoor attacks are performed by embedding the hidden instructions into machine learning models, so that the infected model performs well on benign testing samples when the backdoor is not activated, while its prediction will be changed to the attacker-specified target label when the backdoor is activated by the attacker [94].

In order to defend against these data attacks or model attacks, the malicious users should be identified by analyzing the updated intermediate data using dimensionality reduction methods, e.g., principal component analysis (PCA) [165], anomaly detection [90,101], or interpretability techniques [13]. In addition, the model poisoning can be incurred by Byzantine failures of certain distributed computing resources [40]. With Byzantine failures, some computing resources (bad users) are manipulated by attackers during the distributed training process, which significantly degrades the performance of the global model in terms of test error [40]. In order to make the training process robust against the Byzantine failures, the bad users can be identified by analyzing the updated intermediate data using a hidden Markov model [39,130] or via secure aggregation protocols [70].

5 Federated learning frameworks

FL systems are widely applied in diverse domains, e.g., mobile service, healthcare [188], and finance [89]. An FL system generally exploits an FL framework, which is deployed on

distributed resources. In this section, we present four widely used FL frameworks: PaddleFL [10], TensorFlowFederated [55], FATE [182], and PySyft [136].

5.1 PaddleFL

PaddleFL is an open-source federated learning framework based on PaddlePaddle [116], which is supported by Baidu. At the presentation layer, PaddleFL provides a textual UI for the interaction between users and the FL system. At the user services layer, PaddleFL provides the log and monitoring supports, and it can leverage the interpretability module [11] of PaddlePaddle in the future. At the FL training layer, PaddleFL can realize data parallelism (horizontal FL) and model parallelism (vertical FL). It supports multiple aggregation algorithms, e.g., FedAvg, and fault-tolerance. At the infrastructure layer, PaddleFL exploits RPC for the distributed execution. PaddleFL exploits DP to protect the data security. PaddleFL is widely used in multiple domains, e.g., natural language processing (NLP), computing vision (CV) [109], and recommendation.

5.2 TensorFlowFederated

TensorFlow Federated (TFF) [55] is an open-source framework for federated learning on decentralized data, which is supported by Google. TFF also provides a textual UI through Python. TFF supports the monitoring and log functionality at the user service layer. TFF supports data parallelism (horizontal FL), multiple aggregation algorithms, and fault-tolerance of mobile devices. TFF exploits RPC for the distributed execution and DP for the protection of data privacy. TFF enables Android mobile users to predict the next word while using the keyboard on their mobile phones [120,122].

5.3 FATE

FATE [182] is an open-source FL framework supported by WeBank. FATE provides both a graphical and textual UI. FATE can support the monitoring of distributed training through a web portal. FATE takes advantage of database management systems (DBMS) to track the execution status. FATE can enable horizontal (data parallelism), vertical (model parallelism), and hybrid federated learning. FATE exploits both the DP and HE to protect the data privacy. In addition, FATE exploits RPC to perform the distributed execution.

5.4 PySyft

PySyft [148] is an open-source FL framework based on the PyTorch framework [144]. PySyft is written in Python and provides a textual UI based on Python. PySyft mainly supports the data parallelism and model parallelism based on an aggregator or orchestrating server. The aggregator or orchestrating server sends a part of the model to participating clients to process local data and gets results for federated averaging. PySyft exploits DP and encryption techniques to protect the data security. PySyft exploits multiple communication protocols for distributed execution, e.g., RPC, websocket [44], etc.

Table 3 Comparison among diverse frameworks

Framework	Engine	Aggregation	UI	Parallelism	Security
PaddleFL	Paddle	Centralized	textual	Data/Model/Pipeline	DP/HE
TFF	TensorFlow	Centralized	textual	Data/Model	DP/HE
FATE	TensorFlow	Centralized	Web	Data/Model	DP/HE
PySyft	PyTorch	Centralized	textual	Data	DP/HE

Aggregation represents the type of aggregation algorithms. *Textual* represents the textual UI, while *Web* represents Web portal

Table 4 Comparison among diverse frameworks for the support of diverse FL types, e.g., horizontal FL, vertical FL, and hybrid FL, and GPU

		PaddleFL	TFF	FATE	PySyft
Types	Horizontal	✓	✓	✓	✓
	Vertical	✓	✗	✓	‖✓‖
	Hybrid	✓	✗	✓	✗
GPU		✓	✓	✓	✓

‖✓‖ represents that the support is not realized by itself but a close one

5.5 Concluding remarks

Diverse FL frameworks exist while each has its advantage. We summarize the characteristics of each framework in Table 3, so as to help select a proper framework for use. From the table, we can see that all the frameworks implement the centralized aggregation algorithms, while employing DP and HE for the data security. PaddleFL can exploit Paddle to realize data, model, and pipeline parallelism. FATE and TFF are based on Tensorflow as the engine, while FATE can provide Web portal UI, which is convenient for novices. PySyft is compatible with PyTorch, which can easily handle the PyTorch-based tasks, while PaddleFL is compatible with Paddle, which can easily deal with rich pre-trained models published in PaddleHub [137].

Table 4 represents the support of diverse types of FL in terms of data distribution. All the frameworks support horizontal FL, while vertical FL is supported by three frameworks except TFF. PySyft cannot directly support the vertical FL, while PyVertical [146], which is built upon PySyft, can be used to support vertical FL with the compatibility of PyTorch models. The hybrid FL is only supported by Paddle and FATE. In addition, all the frameworks support the execution with GPU. In practice, although PaddleFL may correspond to slightly longer time, the accuracy of the trained model can be higher that of TFF and FATE, while PySyft may generate "out of memory" errors [85].

6 Research directions

Although much work has been done on the FL systems, there remain some limitations, e.g., interpretability of FL, decentralized aggregation, FL on graphs, benchmarks of FL systems, and applications to distributed intelligent systems. This section discusses the limitations of the existing frameworks and proposes new research directions.

6.1 Benchmarks

Several datasets exist for experiments on FL systems. For instance, Federated Extended MNIST (FEMNIST) [21] is built by partitioning the data in Extended MNIST [32] based on each writer. Shakespeare [120] is built from The Complete Works of William Shakespeare [155] based on each speaking role. Both of these datasets can be used for horizontal FL. However, no public datasets exist for vertical FL or transfer FL. In addition, no open decentralized IID or non-IID distribution of popular datasets, e.g., ImageNet [35], exist for FL systems.

6.2 Interpretability

Deep neural networks have excellent performance in various areas, while it is often difficult to understand the results of deep neural network models, especially within FL systems. Shapley values have been used to provide the interpretability [175], while it focuses on vertical FL. When multiple users collaboratively train an FL model, it remains an open problem to evaluate the contributions of each user, which helps provide evidence for the incentive of each user. The primary incentive for clients to participate in federated learning is obtaining better models [87], while the benefit of participating in federated learning for clients who have sufficient private data to train accurate local models is disputable. Interpretability can help understand the contributions of each user and provide an objective opinion on the incentive strategy within an FL system. In addition, the interpretability helps domain experts to understand the relationship between data and the final trained model in critical domains, e.g., healthcare and finance. However, the interpretability within FL systems remains an open problem.

6.3 Decentralized aggregation

Current aggregation algorithms of FL systems focus on the full connection or star connection topology, while other topologies, e.g., dynamic exponential-2 graph, may help accelerate the distributed training with FL systems [96]. In addition, well-known graph algorithms, e.g., graph partitioning algorithms, and ad hoc policies can be exploited to help better distribute computing resources with the topology defined in Sect. 3.2.3 in order to improve the efficiency of FL systems. While the peer-to-peer communication enables the FL with an arbitrary topology matrix, the data security under diverse attacks, e.g., data or model poisoning, GAN-based attacks, remain open problems and deserve further investigation.

6.4 Federated learning on graphs

Graphs or graph neural networks (GNNs) [169] have gained increasing popularity in multiple domains, e.g., social network, knowledge graph, and recommender system. FL frameworks for graphs, i.e., GraphFL [173], and GNN, i.e., SGNN [123], have been proposed to train a model with decentralized graphs. However, the data security of FL on graphs remains an open problem. In addition, while a multimodal knowledge graph could not only contain text but also images or other type of data [202], it is worth further exploration to efficiently support the multimodal knowledge graph construction within an FL system [102].

6.5 Imbalanced data

Although FL focuses on the non-IID data, the real-world decentralized data usually exhibit an imbalanced distribution [69,186]. While the imbalanced data exist in multiple areas, such as computer vision [135], bioinformatics, and biomedicine [195], learning from such data requires special attention upon data sampling [195,197], data augmentation [135], and loss function designs [179]. The imbalanced data is related to diverse tasks, e.g., two-class or multi-class classification [14,196]. However, an optimized approach can be proposed to address the imbalanced data within FL systems.

6.6 Applications to distributed intelligent systems

Machine learning algorithms have been widely used to boost the performance of intelligent systems, while FL systems could further enhance intelligent systems [103] in distributed computing environments [104,143] with privacy and security ensured. An intelligent system is a group of machines that has the capacity to gather data, analyze the data, and respond to other systems or the world around. With FL systems, the distributed data can be exploited to generate models of high performance so as to produce smart responses.

7 Conclusion

In this paper, we discussed the current state of the art of FL systems, including the functional architecture of FL systems, distributed training, and data manipulation.

First, we presented an overview of FL systems. In particular, we introduced the life cycle of FL models, including four phases. Then, we presented the four-layer functional architecture of FL systems, including presentation, user services, FL training, and infrastructure, and we presented each layer in detail.

Second, we detailed the distributed training with two parts, i.e., parallelism and aggregation algorithms. We presented three types of parallelism, including data parallelism, model parallelism, and pipeline parallelism. We associate each parallelism to a corresponding type of FL. For instance, data parallelism is associated with the horizontal FL, which corresponds to cross-device or cross-silo FL. Model parallelism is related to vertical FL and cross-silo FL. We presented the features of different aggregation algorithms in three types: centralized aggregation, hierarchical aggregation, and decentralized aggregation.

Third, we presented the techniques for data manipulation within FL systems. We showed that FL systems prefer RPC for the distributed execution, to handle the fault-tolerance because of moderate network connection. Intermediate data are sketched in order to compress the data, so as to reduce the data communication time. In addition, we presented the data privacy and model security attacks and corresponding defense techniques, e.g., DP, HE, TEE, and the analysis of updated intermediate data for malicious user identification.

We mainly introduced four FL systems: PaddleFL, TensorFlowFederated, FATE, and PySyft. The current solutions primarily focus on the horizontal FL. And we identified five research directions that deserve further investigation: benchmarks, interpretability, decentralized aggregation, FL on graphs, imbalanced data, and the applications of FL systems to distributed intelligent systems.

References

1. Abad MS, Ozfatura E, Gunduz D, Ercetin O (2020) Hierarchical federated learning across heterogeneous cellular networks. In: IEEE int. conf. on acoustics, speech and signal processing (ICASSP), pp 8866–8870
2. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: ACM SIGSAC conf. on computer and communications security, pp 308–318
3. Abou El Houda Z, Hafid A, Khoukhi L (2019) Co-IOT: a collaborative DDOS mitigation scheme in IOT environment based on blockchain using SDN. In: IEEE global communications conference (GLOBECOM), pp 1–6
4. Aono Y, Hayashi T, Wang L, Moriai S (2017) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forens Secur 13(5):1333–1345
5. Arivazhagan MG, Aggarwal V, Singh AK, Choudhary S (2019) Federated learning with personalization layers. ArXiv preprint arXiv:1912.00818
6. Assran M, Loizou N, Ballas N, Rabbat M (2019) Stochastic gradient push for distributed deep learning. Int Confer Mach Learn 97:344–353
7. Ateniese G, Mancini LV, Spognardi A, Villani A, Vitali D, Felici G (2015) Hacking smart machines with smarter ones: how to extract meaningful data from machine learning classifiers. Int J Secur Netw 10(3):137–150
8. Awan AA, Chu CH, Subramoni H, Panda DK (2018) Optimized broadcast for deep learning workloads on dense-GPU infiniband clusters: MPI or NCCL? In: European MPI Users' Group Meeting, pp 1–9
9. Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V (2020) How to backdoor federated learning. In: Int. conf. on artificial intelligence and statistics (AISTATS), pp 2938–2948
10. Baidu. Federated deep learning in paddlepaddle (online). https://github.com/PaddlePaddle/PaddleFL. Accessed 16 Feb 2021
11. Baidu. Paddlepaddle interpretability. https://github.com/PaddlePaddle/InterpretDL (online). Accessed 13 Mar 2021
12. Beutel DJ, Topal T, Mathur A, Qiu X, Parcollet T, de Gusmão PP, Lane ND (2020) Flower: a friendly federated learning research framework. ArXiv preprint arXiv:2007.14390
13. Bhagoji AN, Chakraborty S, Mittal P, Calo S (2019) Analyzing federated learning through an adversarial lens. In: Int. conf. on machine learning (ICML), pp 634–643
14. Bi J, Zhang C (2018) An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme. Knowl-Based Syst 158:81–93
15. Bian J, Xiong H, Cheng W, Hu W, Guo Z, Fu Y (2017) Multi-party sparse discriminant learning. In: 2017 IEEE international conference on data mining (ICDM). IEEE, pp 745–750
16. Bian J, Xiong H, Fu Y, Huan J, Guo Z (2020) Mp2sda: multi-party parallelized sparse discriminant learning. ACM Trans Knowl Discov Data (TKDD) 14(3):1–22
17. Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, Kiddon C, Konečnỳ J, Mazzocchi S, McMahan B, Van Overveldt T (2019) Towards federated learning at scale: system design. In: Machine learning and systems (MLSys)
18. Briggs C, Fan Z, Andras P (2020) Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In: Int. joint conf. on neural networks (IJCNN). IEEE, pp 1–9
19. Brisimi TS, Chen R, Mela T, Olshevsky A, Paschalidis IC, Shi W (2018) Federated learning of predictive models from federated electronic health records. Int J Med Inf 112:59–67
20. California consumer privacy act home page (online). Californians for Consumer Privacy. https://www.caprivacy.org/. Accessed 14 Feb 2021
21. Caldas S, Duddu SM, Wu P, Li T, Konečnỳ J, McMahan HB, Smith V, Talwalkar A (2018) Leaf: a benchmark for federated settings. ArXiv preprint arXiv:1812.01097
22. Caldas S, Konečnỳ J, McMahan HB, Talwalkar A (2018) Expanding the reach of federated learning by reducing client resource requirements. ArXiv preprint arXiv:1812.07210
23. Canini K, Chandra T, Ie E, McFadden J, Goldman K, Gunter M, Harmsen J, LeFevre K, Lepikhin D, Llinares TL, Mukherjee I (2012) Sibyl: a system for large scale supervised machine learning. Techn Talk 1:113
24. Çatak FÖ (2015) Secure multi-party computation based privacy preserving extreme learning machine algorithm over vertically distributed data. In: Int. conf. on neural information processing (ICONIP), pp 337–345
25. Chatterjee S, Seneta E (1977) Towards consensus: some convergence theorems on repeated averaging. J Appl Probab 14(1):89–97
26. Chen CL, Golubchik L, Paolieri M (2020) Backdoor attacks on federated meta-learning. ArXiv preprint arXiv:2006.07026

27. Chen J, Sayed AH (2012) Diffusion adaptation strategies for distributed optimization and learning over networks. IEEE Trans Signal Process 60(8):4289–4305
28. Chen M, Zhang W, Yuan Z, Jia Y, Chen H (2020) Fede: embedding knowledge graphs in federated setting. ArXiv preprint arXiv:2010.12882
29. Chen Y, Sun X, Jin Y (2019) Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. IEEE Trans Neural Netw Learn Syst 31(10):4229–4238
30. Chen Y, Luo F, Li T, Xiang T, Liu Z, Li J (2020) A training-integrity privacy-preserving federated learning scheme with trusted execution environment. Inf Sci 522:69–79
31. Chik WB (2013) The Singapore personal data protection act and an assessment of future trends in data privacy reform. Comput Law Secur Rev 29(5):554–575
32. Cohen G, Afshar S, Tapson J, Van Schaik A (2017) Emnist: extending mnist to handwritten letters. In Int. joint conf. on neural networks (IJCNN), pp 2921–2926
33. Conger K. Uber settles data breach investigation for $148 million. https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html (Online). Accessed 17 Feb 2021
34. Conger K (2018) Uber settles data breach investigation for $148 million. https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html (online). Accessed 28 Feb 2021
35. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: *IEEE conf. on computer vision and pattern recognition (CVPR)*, pp. 248–255
36. Deng Y, Kamani MM, Mahdavi M (2020) Adaptive personalized federated learning. ArXiv preprint arXiv:2003.13461
37. Dinh CT, Tran N, Nguyen J (2020) Personalized federated learning with Moreau envelopes. ArXiv preprint arXiv:2006.08848
38. Dwork C (2008) Differential privacy: a survey of results. In: Int. conf. on theory and applications of models of computation, pp 1–19
39. Eddy SR (2004) What is a hidden Markov model? Nat Biotechnol 22(10):1315–1316
40. Fang M, Cao X, Jia J, Gong N (2020) Local model poisoning attacks to byzantine-robust federated learning. In: USENIX security symposium (USENIX security), pp 1605–1622
41. Ferhat ÖÇ, Mustacoglu AF (2018) CPP-ELM: cryptographically privacy-preserving extreme learning machine for cloud systems. Int J Comput Intell Syst 11(1):33–44
42. Feng S, Yu H (2020) Multi-participant multi-class vertical federated learning. ArXiv preprint arXiv:2001.11154
43. Feng Z, Xiong H, Song C, Yang S, Zhao B, Wang L, Chen Z, Yang S, Liu L, Huan J (2019) Securegbm: secure multi-party gradient boosting. In IEEE int. conf. on big data (big data), pp 1312–1321
44. Fette I, Melnikov A (2011) The websocket protocol. RFC, 6455:1–71
45. Flynn MJ (1972) Some computer organizations and their effectiveness. IEEE Trans Comput 100(9):948–960
46. Fung C, Yoon CJ, Beschastnikh I (2018) Mitigating sybils in federated learning poisoning. ArXiv preprint arXiv:1808.04866
47. Gaff BM, Sussman HE, Geetter J (2014) Privacy and big data. Computer 47(6):7–9
48. Ganga K, Karthik S (2013) A fault tolerent approach in scientific workflow systems based on cloud computing. In: Int. conf. on pattern recognition, informatics and mobile engineering, pp 387–390
49. Geiping J, Bauermeister H, Dröge H, Moeller M (2020) Inverting gradients—How easy is it to break privacy in federated learning? ArXiv preprint arXiv:2003.14053
50. Geyer RC, Klein T, Nabi M (2017) Differentially private federated learning: a client level perspective. ArXiv preprint arXiv:1712.07557
51. Gibiansky A (2017) Bringing HPC techniques to deep learning. https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/ (online). Accessed 12 Aug 2020
52. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L (2018) Explaining explanations: an overview of interpretability of machine learning. In: IEEE int. conf. on data science and advanced analytics (DSAA). IEEE, pp 80–89
53. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L (2016) Deep learning, vol 1. MIT Press, Cambridge
54. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: Int. conf. on learning representations (ICLR)
55. Google. Tensorflow federated: machine learning on decentralized data. https://www.tensorflow.org/federated (online). Accessed 16 Feb 2021
56. Gropp W, Gropp WD, Lusk E, Skjellum A, Lusk AD (1999) Using MPI: portable parallel programming with the message-passing interface, vol 1. MIT Press

57. Haddadpour F, Kamani MM, Mokhtari A, Mahdavi M (2020) Federated learning with compression: unified analysis and sharp guarantees. ArXiv preprint arXiv:2007.01154
58. Hao M, Li H, Xu G, Liu S, Yang H (2019) Towards efficient and privacy-preserving federated deep learning. In: IEEE int. conf. on communications (ICC), pp 1–6
59. Hardy S, Henecka W, Ivey-Law H, Nock R, Patrini G, Smith G, Thorne B (2017) Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. ArXiv preprint arXiv:1711.10677
60. He C, Avestimehr S, Annavaram M (2020) Group knowledge transfer: collaborative training of large CNNs on the edge. ArXiv preprint arXiv:2007.14513
61. He C, Annavaram M, Avestimehr S (2020) Towards non-IID and invisible data with FEDNAS: federated deep learning via neural architecture search. ArXiv preprint arXiv:2004.08546
62. He C, Balasubramanian K, Ceyani E, Yang C, Xie H, Sun L, He L, Yang L, Yu PS, Rong Y, Zhao P (2021) Fedgraphnn: a federated learning system and benchmark for graph neural networks. ArXiv preprint arXiv:2104.07145
63. He C, Ceyani E, Balasubramanian K, Annavaram M, Avestimehr S (2021) Spreadgnn: serverless multi-task federated learning for graph neural networks. ArXiv preprint arXiv:2106.02743
64. He C, Li S, Soltanolkotabi M, Avestimehr S (2021) Pipetransformer: automated elastic pipelining for distributed training of large-scale models. In: Int. conf. on machine learning, volume 139 of machine learning research, pp 4150–4159
65. He C, Li S, So J, Zeng X, Zhang M, Wang H, Wang X, Vepakomma P, Singh A, Qiu H, Zhu X (2020) Fedml: a research library and benchmark for federated machine learning. ArXiv preprint arXiv:2007.13518
66. He C, Shah AD, Tang Z, Sivashunmugam DF, Bhogaraju K, Shimpi M, Shen L, Chu X, Soltanolkotabi M, Avestimehr S. Fedcv: a federated learning framework for diverse computer vision tasks. ArXiv preprint arXiv: 2111.11066
67. He C, Tan C, Tang H, Qiu S, Liu J (2019) Central server free federated learning over single-sided trust social networks. ArXiv preprint arXiv:1910.04956
68. He C, Ye H, Shen L, Zhang T (2020) Milenas: efficient neural architecture search via mixed-level reformulation. In: IEEE/CVF conf. on computer vision and pattern recognition (CVPR)
69. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
70. He L, Karimireddy SP, Jaggi M (2020) Secure byzantine-robust machine learning. ArXiv preprint arXiv:2006.04747
71. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: information leakage from collaborative deep learning. In: ACM SIGSAC conference on computer and communications security, pp 603–618
72. Hu C, Jiang J, Wang Z (2019) Decentralized federated learning: a segmented gossip approach. ArXiv preprint arXiv:1908.07782
73. Hu Z, Shaloudegi K, Zhang G, Yu Y (2020) Fedmgda+: federated learning meets multi-objective optimization. ArXiv preprint arXiv:2006.11489
74. Huang Y, Cheng Y, Bapna A, Firat O, Chen D, Chen M, Lee H, Ngiam J, Le QV, Wu Y (2018) Gpipe: efficient training of giant neural networks using pipeline parallelism. ArXiv preprint arXiv:1811.06965
75. Ivkin N, Rothchild D, Ullah E, Stoica I, Arora R (2019) Communication-efficient distributed SGD with sketching. ArXiv preprint arXiv:1903.04488
76. Jiang J, Fu F, Yang T, Cui B (2018) Sketchml: accelerating distributed machine learning with data sketches. In: Int. conf. on management of data, pp 1269–1284
77. Jiang J, Ji S, Long G (2020) Decentralized knowledge acquisition for mobile internet applications. In: World Wide Web, pp 1–17
78. Jiang M, Jung T, Karl R, Zhao T (2020) Federated dynamic GNN with secure aggregation. ArXiv preprint arXiv:2009.07351
79. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RG (2019) Advances and open problems in federated learning. ArXiv preprint arXiv:1912.04977
80. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RG, et al. (2021) Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1)
81. Karimireddy SP, Kale S, Mohri M, Reddi S, Stich S, Suresh AT (2020) Scaffold: stochastic controlled averaging for federated learning. In: Int. conf. on machine learning (ICML), pp 5132–5143
82. Karimireddy SP, Rebjock Q, Stich S, Jaggi M (2019) Error feedback fixes signsgd and other gradient compression schemes. In: Int. conf. on machine learning (ICML), pp 3252–3261

83. Katevas K, Bagdasaryan E, Waterman J, Safadieh MM, Birrell E, Haddadi H, Estrin D (2020) Policy-based federated learning. ArXiv e-prints, pp arXiv-2003
84. Ke C, Honorio J (2021) Federated myopic community detection with one-shot communication. ArXiv preprint arXiv:2106.07255
85. Kholod I, Yanaki E, Fomichev D, Shalugin E, Novikova E, Filippov E, Nordlund M (2021) Open-source federated learning frameworks for iot: a comparative review and analysis. Sensors 21(1):167
86. Konečnỳ J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: strategies for improving communication efficiency. ArXiv preprint arXiv:1610.05492
87. Kulkarni V, Kulkarni M, Pant A (2020) Survey of personalization techniques for federated learning. In: World conf. on smart trends in systems, security and sustainability (WorldS4), pp 794–797
88. Lalitha A, Kilinc OC, Javidi T, Koushanfar F (2019) Peer-to-peer federated learning on graphs. ArXiv preprint arXiv:1901.11173
89. Li Q, Wen Z, He B (2019) A survey on federated learning systems: vision, hype and reality for data privacy and protection. ArXiv preprint arXiv:1907.09693
90. Li S, Cheng Y, Liu Y, Wang W, Chen T (2019) Abnormal client behavior detection in federated learning. ArXiv preprint arXiv:1910.09933
91. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
92. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V (2020) Federated optimization in heterogeneous networks. Mach Learn Syst 2:429–450
93. Li T, Sanjabi M, Beirami A, Smith V (2019) Fair resource allocation in federated learning. arXiv preprint arXiv:1905.10497
94. Li Y, Wu B, Jiang Y, Li Z, Xia ST (2020) Backdoor learning: a survey. arXiv preprint arXiv:2007.08745
95. Li Z, Huang Z, Chen C, Hong C (2019) Quantification of the leakage in federated learning. arXiv preprint arXiv:1910.05467
96. Lian X, Zhang C, Zhang H, Hsieh CJ, Zhang W, Liu J (2017) Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Advances in neural information processing systems (NeurIPS), pp 5330–5340
97. Liang Z, Wang B, Gu Q, Osher S, Yao Y (2020) Exploring private federated learning with laplacian smoothing. arXiv preprint arXiv:2005.00218
98. Liaqat M, Chang V, Gani A, Ab Hamid SH, Toseef M, Shoaib U, Ali RL (2017) Federated cloud resource management: review and discussion. J Netw Comput Appl 77:87–105
99. Lim WY, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, Niyato D, Miao C (2020) Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun Surv Tutor 22(3):2031–2063
100. Lin BY, He C, Zeng Z, Wang H, Huang Y, Soltanolkotabi M, Ren X, Avestimehr S (2021) Fednlp: a research platform for federated learning in natural language processing. ArXiv preprint arXiv:2104.08815
101. Lin J, Du M, Liu J (2019) Free-riders in federated learning: attacks and defenses. ArXiv preprint arXiv:1911.12560
102. Lin Y, Chen C, Chen C, Wang L (2020) Improving federated relational data modeling via basis alignment and weight penalty. ArXiv preprint arXiv:2011.11369
103. Liu J, Bondiombouy C, Mo L, Valduriez P (2020) Two-phase scheduling for efficient vehicle sharing. IEEE Trans Intell Transp Syst (TITS) 23(1): 457–470
104. Liu J, Pacitti E, Valduriez P, De Oliveira D, Mattoso M (2016) Multi-objective scheduling of scientific workflows in multisite clouds. Fut Gener Comput Syst 63:76–95
105. Liu J, Pacitti E, Valduriez P, Mattoso M (2015) A survey of data-intensive scientific workflow management. J Grid Comput 13(4):457–493
106. Liu J, Pineda L, Pacitti E, Costan A, Valduriez P, Antoniu G, Mattoso M (2018) Efficient scheduling of scientific workflows using hot metadata in a multisite cloud. IEEE Trans Knowl Data Eng 31(10):1940–1953
107. Liu L, Zhang J, Song SH, Letaief KB (2020) Client-edge-cloud hierarchical federated learning. In: IEEE int. conf. on communications (ICC), pp 1–6
108. Liu R, Cao Y, Yoshikawa M, Chen H (2020) Fedsel: federated sgd under local differential privacy with top-k dimension selection. In: Int. conf. on database systems for advanced applications, pp 485–501
109. Liu Y, Huang A, Luo Y, Huang H, Liu Y, Chen Y, Feng L, Chen T, Yu H, Yang Q (2020) Fedvision: an online visual object detection platform powered by federated learning. AAAI Confer Artif Intell 34:13172–13179
110. Liu Y, Kang Y, Zhang X, Li L, Cheng Y, Chen T, Hong M, Yang Q (2019) A communication efficient collaborative learning framework for distributed features. ArXiv preprint arXiv:1912.11187

111. Lo SK, Lu Q, Zhu L, Paik HY, Xu X, Wang C (2021) Architectural patterns for the design of federated learning systems. ArXiv preprint arXiv:2101.02373
112. Luo S, Chen X, Wu Q, Zhou Z, Yu S (2020) Hfel: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. IEEE Trans Wirel Commun 19(10):6535–6548
113. Luo X, Zhu X (2020) Exploiting defenses against gan-based feature inference attacks in federated learning. ArXiv preprint arXiv:2004.12571
114. Lyu L, Yu H, Yang Q (2020) Threats to federated learning: a survey. ArXiv preprint arXiv:2003.02133
115. Lyu L, Yu J, Nandakumar K, Li Y, Ma X, Jin J, Yu H, Ng KS (2020) Towards fair and privacy-preserving federated deep models. IEEE Trans Parallel Distrib Syst 31(11):2524–2541
116. Ma Y, Yu D, Wu T, Wang H (2019) Paddlepaddle: an open-source deep learning platform from industrial practice. Front Data Comput 1(1):105
117. Malekijoo A, Fadaeieslam MJ, Malekijou H, Homayounfar M, Alizadeh-Shabdiz F, Rawassizadeh R (2021) FEDZIP: a compression framework for communication-efficient federated learning. ArXiv preprint arXiv:2102.01593
118. Mandal K, Gong G (2019) PrivFL: practical privacy-preserving federated regressions on high-dimensional data over mobile networks. In: ACM SIGSAC conf. on cloud computing security workshop, pp 57–68
119. McKeen F, Alexandrovich I, Berenzon A, Rozas CV, Shafi H, Shanbhogue V, Savagaonkar UR (2013) Innovative instructions and software model for isolated execution. In: Int. workshop on hardware and architectural support for security and privacy
120. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Int. conf. on artificial intelligence and statistics (AISTATS), pp 1273–1282
121. McMahan HB, Ramage D, Talwar K, Zhang L (2017) Learning differentially private recurrent language models. ArXiv preprint arXiv:1710.06963
122. McMahan HB, Ramage D, Talwar K, Zhang L (2018) Learning differentially private recurrent language models. In: Int. conf. on learning representations (ICLR)
123. Mei G, Guo Z, Liu S, Pan L (2019) Sgnn: a graph neural network based federated learning approach by hiding structure. In: IEEE int. conf. on big data (big data), pp 2560–2568
124. Melis L, Song C, De Cristofaro E, Shmatikov V (2019) Exploiting unintended feature leakage in collaborative learning. In: IEEE symposium on security and privacy (SP), pp 691–706
125. Meng C, Rambhatla S, Liu Y (2021) Cross-node federated graph neural network for spatio-temporal data modeling. In: ACM SIGKDD conference on knowledge discovery and data mining (KDD) (to appear)
126. Mhaisen N, Abdellatif AA, Mohamed A, Erbad A, Guizani M (2021) Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints. IEEE Trans Netw Sci Eng 9(1): 55–66
127. Mo F, Haddadi H (2019) Efficient and private federated learning using tee. In: EuroSys
128. Mohri M, Sivek G, Suresh AT (2019) Agnostic federated learning. In: Int. conf. on machine learning (ICML), pp 4615–4625
129. Mothukuri V, Parizi RM, Pouriyeh S, Huang Y, Dehghantanha A, Srivastava G (2021) A survey on security and privacy of federated learning. Fut Gener Comput Syst 115:619–640
130. Muñoz-González L, Co KT, Lupu EC (2019) Byzantine-robust federated machine learning through adaptive model averaging. ArXiv preprint arXiv:1909.05125
131. Narayanan D, Harlap A, Phanishayee A, Seshadri V, Devanur NR, Ganger GR, Gibbons PB, Zaharia M (2019) Pipedream: generalized pipeline parallelism for dnn training. In: ACM symposium on operating systems principles, pp 1–15
132. Ochiai K, Senkawa K, Yamamoto N, Tanaka Y, Fukazawa Y (2019) Real-time on-device troubleshooting recommendation for smartphones. In: ACM SIGKDD int. conf. on knowledge discovery and data mining, pp 2783–2791
133. Official Journal of the European Union. General data protection regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (online). Accessed 12 Feb 2021
134. Ohrimenko O, Schuster F, Fournet C, Mehta A, Nowozin S, Vaswani K, Costa M (2016) Oblivious multi-party machine learning on trusted processors. In {USENIX} security symposium ({USENIX} security), pp 619–636
135. Oksuz K, Cam BC, Kalkan S, Akbas E (2020) Imbalance problems in object detection: a review. IEEE Trans Pattern Anal Mach Intell 43:3388–3415
136. OpenMined. Pysyft. https://github.com/OpenMined/PySyft (online). Accessed 22 Feb 2021
137. PaddlePaddle B. Paddlehub. https://github.com/PaddlePaddle/PaddleHub (online). Accessed 01 Oct 2021

138. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Int. conf. on the theory and applications of cryptographic techniques, pp 223–238
139. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
140. Peng H, Li H, Song Y, Zheng V, Li J (2021) Federated knowledge graphs embedding. In: ACM int. conf. on information and knowledge management (CIKM), pp 1–10
141. Phan H, Thai MT, Hu H, Jin R, Sun T, Dou D (2020) Scalable differential privacy with certified robustness in adversarial learning. In: Int. conf. on machine learning (ICML), pp 7683–7694
142. Pillutla K, Kakade SM, Harchaoui Z (2019) Robust aggregation for federated learning. ArXiv preprint arXiv:1912.13445
143. Pineda-Morales L, Liu J, Costan A, Pacitti E, Antoniu G, Valduriez P, Mattoso M (2016) Managing hot metadata for scientific workflows on multisite clouds. In: IEEE Int. Conf. on Big Data (Big Data), pp 390–397
144. Pytorch. Pytorch. https://pytorch.org/ (online). Accessed 13 Mar 2021
145. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400–407
146. Romanini D, Hall AJ, Papadopoulos P, Titcombe T, Ismail A, Cebere T, Sandmann R, Roehm R, Hoeh MA (2021) Pyvertical: a vertical federated learning framework for multi-headed splitnn. ArXiv preprint arXiv:2104.00489
147. Rothchild D, Panda A, Ullah E, Ivkin N, Stoica I, Braverman V, Gonzalez J, Arora R (2020) Fetchsgd: communication-efficient federated learning with sketching. In: Int. conf. on machine learning (ICML), pp 8253–8265
148. Ryffel T, Trask A, Dahl M, Wagner B, Mancuso J, Rueckert D, Passerat-Palmbach J (2018) A generic framework for privacy preserving deep learning. ArXiv preprint arXiv:1811.04017
149. Sabater C, Bellet A, Ramon J (2020) Distributed differentially private averaging with improved utility and robustness to malicious parties. ArXiv preprint arXiv:2006.07218
150. Satariano A. Google is fined $57 million under Europe's data privacy law. https://www.nytimes.com/2019/01/21/technology/google-europe-gdpr-fine.html (online). Accessed 28 Feb 2021
151. Sayed AH (2014) Adaptation, learning, and optimization over networks. Found Trends Mach Learn 7(ARTICLE):311–801
152. Sayed AH, Tu SY, Chen J, Zhao X, Towfic ZJ (2013) Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior. IEEE Signal Process Maga 30(3):155–171
153. Seif M, Tandon R, Li M (2020) Wireless federated learning with local differential privacy. In: IEEE int. symposium on information theory (ISIT), pp 2604–2609
154. Seneta E (2006) Non-negative matrices and Markov chains. Springer
155. Shakespeare W (2007) The complete works of William Shakespeare. Wordsworth Editions
156. Shlezinger N, Chen M, Eldar YC, Poor HV, Cui S (2020) Federated learning with quantization constraints. In: IEEE int. conf. on acoustics, speech and signal processing (ICASSP), pp 8851–8855
157. Shlezinger N, Chen M, Eldar YC, Poor HV, Cui S (2020) Uveqfed: universal vector quantization for federated learning. IEEE Trans Signal Process 69:500–514
158. Silverstein J. Hundreds of millions of facebook user records were exposed on amazon cloud server. https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/ (online). Accessed 28 Feb 2021
159. Spring R, Kyrillidis A, Mohan V, Shrivastava A (2019) Compressing gradient optimizers via count-sketches. In: Int. conf. on machine learning (ICML), pp 5946–5955
160. Standing Committee of the National People's Congress. Cybersecurity law of the people's Republic of China. https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-cybersecurity-law-peoples-republic-china/ (online). Accessed 22 Feb 2021
161. Stich SU, Cordonnier JB, Jaggi M (2018) Sparsified SGD with memory. In: Advances in neural information processing systems (NeurIPS), vol 31
162. Sun H, Ma X, Hu RQ (2020) Adaptive federated learning with gradient compression in uplink NOMA. IEEE Trans Vehic Technol 69(12):16325–16329
163. Sun Z, Kairouz P, Suresh AT, McMahan HB (2019) Can you really backdoor federated learning? ArXiv preprint arXiv:1911.07963
164. Suzumura T, Zhou Y, Baracaldo N, Ye G, Houck K, Kawahara R, Anwar A, Stavarache LL, Watanabe Y, Loyola P, Klyashtorny D (2019) Towards federated graph learning for collaborative financial crimes detection. ArXiv preprint arXiv:1909.12946
165. Tolpegin V, Truex S, Gursoy ME, Liu L (2020) Data poisoning attacks against federated learning systems. In: European symposium on research in computer security. Springer, pp 480–501
166. Triastcyn A, Faltings B (2020) Federated generative privacy. IEEE Intell Syst 35(4):50–57

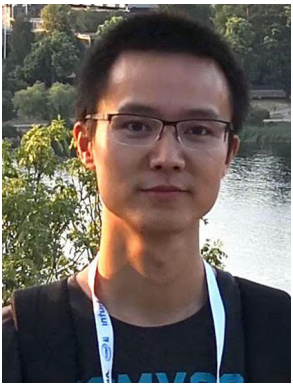
167. Truex Stacey, Baracaldo Nathalie, Anwar Ali, Steinke Thomas, Ludwig Heiko, Zhang Rui, Zhou Yi (2019) A hybrid approach to privacy-preserving federated learning. In: ACM workshop on artificial intelligence and security, pp 1–11
168. Vanhaesebrouck P, Bellet A, Tommasi M (2017) Decentralized collaborative learning of personalized models over networks. In: Int. conf. on artificial intelligence and statistics (AISTATS), pp 509–517
169. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: Int. conf. on learning representations (ICLR)
170. Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS (2020) A survey on distributed machine learning. ACM Comput Surv (CSUR) 53(2):1–33
171. Vishnu A, Siegel C, Daily J (2016) Distributed tensorflow with MPI. ArXiv preprint arXiv:1603.02339
172. Wainakh A, Guinea AS, Grube T, Mühlhäuser M (2020) Enhancing privacy via hierarchical federated learning. In: IEEE European symposium on security and privacy workshops (EuroS&PW), pp 344–347
173. Wang B, Li A, Li H, Chen Y (2020) Graphfl: a federated learning framework for semi-supervised node classification on graphs. ArXiv preprint arXiv:2012.04187
174. Wang C, Chen B, Li G, Wang H (2021) FL-AGCNS: federated learning framework for automatic graph convolutional network search. ArXiv preprint arXiv:2104.04141
175. Wang G (2019) Interpret federated learning with Shapley values. ArXiv preprint arXiv:1905.04519
176. Wang H, Yurochkin M, Sun Y, Papailiopoulos D, Khazaeni Y (2020) Federated learning with matched averaging. In: Int. conf. on learning representations (ICLR)
177. Wang J, Charles Z, Xu Z, Joshi G, McMahan HB, Al-Shedivat M, Andrew G, Avestimehr S, Daly K, Data D, Diggavi S (2021) A field guide to federated optimization. ArXiv preprint arXiv:2107.06917
178. Wang J, Sahu AK, Yang Z, Joshi G, Kar S (2019) Matcha: speeding up decentralized SGD via matching decomposition sampling. In: Indian control conference (ICC), pp 299–300
179. Wang L, Xu S, Wang X, Zhu Q (2021) Addressing class imbalance in federated learning. AAAI Confer Artif Intell 35:10165–10173
180. Luping W, Wei W, Bo L (2019) CMFL: mitigating communication overhead for federated learning. In: IEEE int. conf. on distributed computing systems (ICDCS), pp 954–964
181. Wang Z, Song M, Zhang Z, Song Y, Wang Q, Qi H (2019) Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE conf. on computer communications (INFOCOM), pp 2512–2520
182. WeBank. Federated AI technology enabler (FATE). https://github.com/FederatedAI/FATE (online). Accessed 16 Feb 2021
183. WeBank. Federated learning white paper v2.0. https://aisp-1251170195.cos.ap-hongkong.myqcloud.com/wp-content/uploads/pdf/%E8%81%94%E9%82%A6%E5%AD%A6%E4%B9%A0%E7%99%BD%E7%9A%AE%E4%B9%A6_v2.0.pdf (online). Accessed 14 Feb 2021
184. Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, Jin S, Quek TQ, Poor HV (2020) Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans Inf Forens Secur 15:3454–3469
185. Wu C, Wu F, Cao Y, Huang Y, Xie X (2021) Fedgnn: federated graph neural network for privacy-preserving recommendation. ArXiv preprint arXiv:2102.04925
186. Wu T, Liu Z, Huang Q, Wang Y, Lin D (2021) Adversarial robustness under long-tailed distribution. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8659–8668
187. Xu C, Tao D, Xu C (2013) A survey on multi-view learning. ArXiv preprint arXiv:1304.5634
188. Xu J, Glicksberg BS, Su C, Walker P, Bian J, Wang F (2021) Federated learning for healthcare informatics. J Healthc Inf Res 5:1–19
189. Xu J, Du W, Jin Y, He W, Cheng R (2020) Ternary compression for communication-efficient federated learning. IEEE Trans Neural Netw Learn Syst 33(3):1162–1176
190. Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. ACM Trans Intell Syst Technol 10(2):1–19
191. Yi X, Paulet R, Bertino E (2014) Homomorphic encryption. In: Homomorphic encryption and applications. Springer, pp 27–46
192. Yuan J, Xu M, Ma X, Zhou A, Liu X, Wang S (2020) Hierarchical federated learning through LAN-WAN orchestration. ArXiv preprint arXiv:2010.11612
193. Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaeni Y (2019) Bayesian nonparametric federated learning of neural networks. In: Int. conf. on machine learning (ICML), pp 7252–7261
194. Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y (2020) Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In: USENIX annual technical conference (USENIX ATC), pp 493–506
195. Zhang C, Bi J, Soda P (2017) Feature selection and resampling in class imbalance learning: Which comes first? An empirical study in the biological domain. In: Int. conf. on bioinformatics and biomedicine (BIBM), pp 933–938

196. Zhang C, Bi J, Xu S, Ramentol E, Fan G, Qiao B, Fujita H (2019) Multi-imbalance: an open-source software for multi-class imbalance learning. Knowl-Based Syst 174:137–143
197. Zhang C, Soda P, Bi J, Fan G, Almpanidis G, Garcia S (2021) An empirical study on the joint impact of feature selection and data resampling on imbalance classification. ArXiv preprint arXiv:2109.00201
198. Zhang H, Shen T, Wu F, Yin M, Yang H, Wu C (2021) Federated graph learning—a position paper. ArXiv preprint arXiv:2105.11099
199. Zhang T, He C, Ma T, Gao L, Ma M, Avestimehr S (2021) Federated learning for internet of things: a federated learning framework for on-device anomaly data detection. ArXiv preprint arXiv:2106.07976
200. Zhang X, Li F, Zhang Z, Li Q, Wang C, Wu J (2020) Enabling execution assurance of federated learning at untrusted participants. In: IEEE INFOCOM conf. on computer communications, pp 1877–1886
201. Zhao B, Mopuri KR, Bilen H (2020) IDLG: improved deep leakage from gradients. ArXiv preprint arXiv:2001.02610
202. Zhao Y, Barnaghi P, Haddadi H (2021) Multimodal federated learning. ArXiv preprint arXiv:2109.04833
203. Zheng L, Zhou J, Chen C, Wu B, Wang L, Zhang B (2021) Asfgnn: automated separated-federated graph neural network. Peer-to-Peer Netw Appl 14(3):1692–1704
204. Zhou J, Chen C, Zheng L, Wu H, Wu J, Zheng X, Wu B, Liu Z, Wang L (2020) Vertically federated graph neural network for privacy-preserving node classification. ArXiv preprint arXiv:2005.11903
205. Zhu H, Zhang H, Jin Y (2021) From federated learning to federated neural architecture search: a survey. Complex Intell Syst
206. Zinkevich M, Weimer M, Li L, Smola A. (2010) Parallelized stochastic gradient descent. In: Advances in neural information processing systems (NeurIPS), vol 4. Citeseer, p 4

Ji Liu is a staff researcher with the Big Data Laboratory, Baidu Research, Beijing, China. He was a software engineer at Murex and postdoctoral research fellow with Inria and LIRMM, University of Montpellier, France. Previously, he was a PhD candidate in the Microsoft Research Inria Joint Centre with the Inria Zenith team. His research interests include federated learning, distributed machine learning, scientific workflows, distributed and parallel databases, and distributed system. He has published several papers in top international journals and conferences, such as NeurIPS, AAAI, KDD, TKDD, TITS, and TKDE, and is a co-author of the book "Data-Intensive Workflow Management For Clouds and Data-Intensive and Scalable Computing Environments" published by Morgan Claypool in 2019. He obtained a PhD degree from University of Montpellier in 2016, a master degree from Telecom SudParis in 2013, and a BSc degree from Xidian University in 2011.

Jizhou Huang received his PhD degree from Harbin Institute of Technology, China. He is Chair of Baidu AI Technical Committee and Chief Architect with Baidu Inc. His research fields include data mining, natural language processing, knowledge graph, and other artificial intelligence technologies.

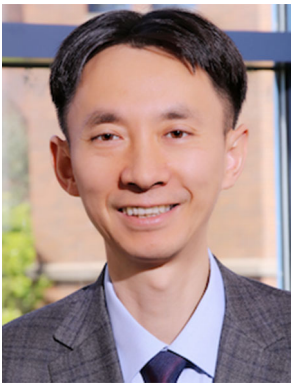
Yang Zhou is an assistant professor in the Department of Computer Science and Software Engineering at the Auburn University. Prior to that, he received his PhD degree in the College of Computing at the Georgia Institute of Technology. His research interests lie in the areas of machine learning and data science, particularly trustworthy machine learning, parallel, distributed, and federated learning, graph machine learning, and natural language processing. His research contributions have been published in top venues of machine learning (ICML, NeurIPS), data mining (KDD, ICDM, TKDD, DMKD, KAIS), artificial intelligence (AAAI, IJCAI, TIST), natural language processing (ACL, EMNLP), Web (WWW, TWEB), high-performance computing (HPDC, SC), database systems (VLDB, TKDE, VLDBJ), networking (JSAC, TOIT), web services (ICWS, TSC), and software engineering (ISSTA).



Xuhong Li is a researcher at Big Data Laboratory of Baidu Researcher. He received his bachelor degree and master degree from Beihang University, and his PhD degree from University of Technology of Compiegne in 2019. He is interested in and working on Explainable AI and Transfer Learning, as well as self-supervised learning and multimodal learning, for both computer vision and natural language processing applications. He has served regularly among the program committees for top international machine learning conferences, such as ICML and Neurips.

Shilei Ji a staff engineer at Baidu, Beijing, China, working on data security and privacy protection for both scientific research and business innovation, with a focus on secure multi-party computation, federated learning, and confidential computing. He obtained a master degree from Tiangong University in 2009.

Haoyi Xiong (Senior Member, IEEE) received the PhD degree in computer science from Telecom SudParis, Université Pierre et Marie Curie, Paris, France, in 2015. From 2016 to 2018, he was a Tenure-Track Assistant Professor with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA (formerly known as University of Missouri at Rolla). From 2015 to 2016, he was a research associate with the Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA, USA. He joined Big Data Laboratory in 2018, where he is currently a Principal R&D Architect and Research Scientist. He also holds an honorary appointment as a Graduate Faculty Scholar affiliated to the ECE PhD Program at University of Central Florida, Orlando FL, USA.



Dejing Dou is the Head of Big Data Lab (BDL) and Business Intelligence Lab (BIL) at Baidu Research. He is also a full professor (on leave) from the Computer and Information Science Department at the University of Oregon. He received his bachelor degree from Tsinghua University, China, in 1996 and his Ph.D. degree from Yale University in 2004. His research areas include artificial intelligence, data mining, data integration, NLP, and health informatics. Dejing Dou has published more than 150 research papers, some of which appear in prestigious conferences and journals like AAAI, IJCAI, ICML, NeurIPS, ICLR, KDD, ICDM, ACL, EMNLP, CVPR, ICCV, CIKM, ISWC, TKDD, JIIS, and JoDS, with more than 4500 Google Scholar citations. His DEXA'15 paper received the best paper award. His KDD'07 paper was nominated for the best research paper award. His COLING'18 paper was Area Chair Favorites (excellent). He is on the Editorial Boards of Journal on Data Semantics, Journal of Intelligent Information Systems, and PLOS ONE. He is a Editor-in-Chief of AIMS Electronic Research Archive. He has been serving as program committee members for major international conferences and as program co-chairs for five of them. He has received over 5 million PI research grants from the NSF and the NIH. Dejing Dou is a senior member of ACM and IEEE.