A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He

Abstract—As data privacy increasingly becomes a critical societal concern, federated learning has been a hot research topic in enabling the collaborative training of machine learning models among different organizations under the privacy restrictions. As researchers try to support more machine learning models with different privacy-preserving approaches, there is a requirement in developing systems and infrastructures to ease the development of various federated learning algorithms. Similar to deep learning systems such as PyTorch and TensorFlow that boost the development of deep learning, federated learning systems (FLSs) are equivalently important, and face challenges from various aspects such as effectiveness, efficiency, and privacy. In this survey, we conduct a comprehensive review on federated learning systems. To understand the key design system components and guide future research, we introduce the definition of federated learning systems and analyze the system components. Moreover, we provide a thorough categorization for federated learning systems according to six different aspects, including data distribution, machine learning model, privacy mechanism, communication architecture, scale of federation and motivation of federation. The categorization can help the design of federated learning systems as shown in our case studies. By systematically summarizing the existing federated learning systems, we present the design factors, case studies, and future research opportunities.

Index Terms—Federated learning, machine learning, data mining, survey

1 Introduction

Many machine learning algorithms are data hungry, and in reality, data are dispersed over different organizations under the protection of privacy restrictions. Due to these factors, federated learning (FL) [77], [119], [191] has become a hot research topic in machine learning. For example, data of different hospitals are isolated and become "data islands". Since each data island has limitations in size and approximating real distributions, a single hospital may not be able to train a high-quality model that has a good predictive accuracy for a specific task. Ideally, hospitals can benefit more if they can collaboratively train a machine learning model on the union of their data. However, the data cannot simply be shared among the hospitals due to various policies and regulations. Such phenomena on "data islands" are commonly seen in many areas such as finance, government, and supply chains. Policies such as General Data Protection Regulation (GDPR) [9] stipulate rules on data sharing among different organizations. Thus, it is challenging to develop a federated learning system which has a good predictive accuracy while obeying policies and regulations to protect privacy.

Many efforts have recently been devoted to implementing federated learning algorithms to support effective machine learning models. Specifically, researchers try to support more machine learning models with different privacy-preserving approaches, including deep neural networks (NNs) [22], [110], [119], [145], [196], gradient boosted decision trees (GBDTs) [36], [95], [200], logistics regression [34], [128] and support vector machines (SVMs) [156]. For instance, Nikolaenko *et al.* [128] and Chen *et al.* [34] propose approaches to conduct FL based on linear regression. Since GBDTs have become very successful in recent years [32], [184], the corresponding Federated Learning Systems (FLSs) have also been proposed by Zhao *et al.* [200], Cheng *et al.* [36], Li *et al.* [95]. Moreover, there are many FLSs supporting the training of NNs. Google proposes a scalable production system which enables tens of millions of devices to train a deep neural network [22].

As there are common methods and building blocks (e.g., privacy mechanisms such as differential privacy) for building FL algorithms, it makes sense to develop systems and infrastructures to ease the development of various FL algorithms. Systems and infrastructures allow algorithm developers to reuse the common building blocks, and avoid building algorithms every time from scratch. Similar to deep learning systems such as PyTorch [135], [136] and TensorFlow [6] that boost the development of deep learning algorithms, FLSs are equivalently important for the success of FL. However, building a successful FLS is challenging, which needs to consider multiple aspects such as effectiveness, efficiency, privacy, and autonomy.

• *Qinbin Li, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He are with the National University of Singapore, Singapore 119077. E-mail: {qinbin, zhaomin, sixuhu, naibowang, liyuan, liuxu, hebs}@comp.nus.edu.sg.*
• *Zeyi Wen is with the The University of Western Australia, Crawley, WA 6009, Australia. E-mail: wenzeyi@gmail.com.*

Manuscript received 19 Jan. 2021; revised 6 Oct. 2021; accepted 27 Oct. 2021. Date of publication 2 Nov. 2021; date of current version 7 Mar. 2023.
This work was supported by National Research Foundation, Singapore under its AI Singapore Programme under Grant AISG2-RP-2020-018.
(Corresponding author: Qinbin Li.)
Recommended for acceptance by L. Chen.
Digital Object Identifier no. 10.1109/TKDE.2021.3124599

In this paper, we take a survey on the existing FLSs from a system view. First, we show the definition of FLSs, and compare it with conventional federated systems. Second, we analyze the system components of FLSs, including the parties, the manager, and the computation-communication framework. Third, we categorize FLSs based on six different aspects: data distribution, machine learning model, privacy mechanism, communication architecture, scale of federation, and motivation of federation. These aspects can direct the design of an FLS as common building blocks and system abstractions. Fourth, based on these aspects, we systematically summarize the existing studies, which can be used to direct the design of FLSs. Last, to make FL more practical and powerful, we present future research directions to work on. We believe that systems and infrastructures are essential for the success of FL. More work has to be carried out to address the system research issues in effectiveness, efficiency, privacy, and autonomy.

1.1 Related Surveys

There have been several surveys on FL. A seminal survey written by Yang *et al* . [191] introduces the basics and concepts in FL, and further proposes a comprehensive secure FL framework. The paper mainly target at a relatively small number of parties which are typically enterprise data owners. Li *et al* . [101] summarize challenges and future directions of FL in massive networks of mobile and edge devices. Recently, Kairouz *et al* . [77] have a comprehensive description about the characteristics and challenges on FL from different research topics. However, they mainly focus on cross-device FL, where the participants are a very large number of mobile or IoT devices. More recently, another survey [10] summarizes the platforms, protocols and applications of federated learning. Some surveys only focus on an aspect of federated learning. For example, Lim *et al* . [105] conduct a survey of FL specific to mobile edge computing, while [116] focuses on the threats to federated learning.

1.2 Our Contribution

To the best of our knowledge, there lacks a survey on reviewing existing systems and infrastructure of FLSs and on boosting the attention of creating systems for FL (Similar to prosperous system research in deep learning). In comparison with the previous surveys, the main contributions of this paper are as follows. (1) Our survey is the first one to provide a comprehensive analysis on FL from a system's point of view, including system components, taxonomy, summary, design, and vision. (2) We provide a comprehensive taxonomy against FLSs on six different aspects, including data distribution, machine learning model, privacy mechanism, communication architecture, scale of federation, and motivation of federation, which can be used as common building blocks and system abstractions of FLSs. (3) We summarize existing typical and state-of-the-art studies according to their domains, which is convenient for researchers and developers to refer to. (4) We present the design factors for a successful FLS and comprehensively review solutions for each scenario. (5) We propose interesting research directions and challenges for future generations of FLSs.

The rest of the paper is organized as follows. In Section 2, we introduce the concept and the system components of FLSs. In Section 3, we propose six aspects to classify FLSs. In Section 4, we summary existing studies and systems on FL. We then present the design factors and solutions for an FLS in Section 5. Last, we propose possible future directions on FL in Section 6 and conclude our paper in Section 7.

2 An Overview of Federated Learning Systems

2.1 Background

As data breach becomes a major concern, more and more governments establish regulations to protect users' data, such as GDPR in European Union [170], PDPA in Singapore [37], and CCPA [1] in the US. The cost of breaching these policies is pretty high for companies. In a breach of 600,000 drivers' personal information in 2016, Uber had to pay $148 million to settle the investigation [3]. SingHealth was fined $750,000 by the Singapore government for a breach of PDPA [5]. Google was fined $57 million for a breach of GDPR [4], which is the largest penalty as of March 18, 2020 under the European Union privacy law.

Under the above circumstances, federated learning, a collaborative learning without exchanging users' original data, has drawn increasingly attention nowadays. While machine learning, especially deep learning, has attracted many attentions again recently, the combination of federation and machine learning is emerging as a new and hot research topic.

2.2 Definition

FL enables multiple parties jointly train a machine learning model without exchanging the local data. It covers the techniques from multiple research areas such as distributed system, machine learning, and privacy. Inspired by the definition of FL given by other studies [77], [191], here we give a definition of FLSs.

In a federated learning system, multiple parties collaboratively train machine learning models without exchanging their raw data. The output of the system is a machine learning model for each party (which can be same or different). A practical federated learning system has the following constraint: given an evaluation metric such as test accuracy, the performance of the model learned by federated learning should be better than the model learned by local training with the same model architecture.

2.3 System Components

There are three major components in an FLS: parties (e.g., clients), the manager (e.g., server), and the communication-computation framework to train the machine learning model.

2.3.1 Parties

In FLSs, the parties are the data owners and the beneficiaries of FL. They can be organizations or mobile devices, named cross-silo or cross-device settings [77], respectively. We consider the following properties of the parties that affect the design of FLSs.

First, what is the hardware capacity of the parties? The hardware capacity includes the computation power and storage. If the parties are mobile phones, the capacity is weak and the parties cannot perform much computation and train a large model. For example, Wang *et al.* [177] consider a resource constrained setting in FL. They design an objective to include the resource budget and proposed an algorithm to determine the rounds of local updates.

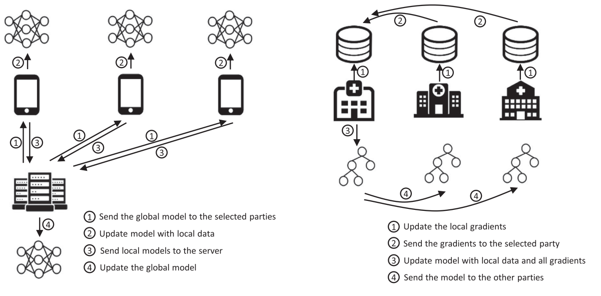
Second, what is the scale and stability of the parties? For organizations, the scale is relative small compared with the mobile devices. Also, the stability of the cross-silo setting is better than the cross-device setting. Thus, in the cross-silo setting, we can expect that every party can continuously conduct computation and communication tasks in the entire federated process, which is a common setting in many studies [36], [95], [156]. If the parties are mobile devices, the system has to handle possible issues such as connection lost [22]. Moreover, since the number of devices can be very large (e.g., millions), it is unpractical to assume all the devices to participate every round in FL. The widely used setting is to choose a fraction of devices to perform computation in each round [22], [119].

Last, what are the data distributions among the parties? Usually, no matter cross-device or cross-silo setting, the non-IID (identically and independently distributed) data distribution is considered a practical and challenging setting in federated learning [77], which is evaluated in the experiments of recent work [95], [103], [174], [196]. Such non-IID data distribution may be more obvious among the organizations. For example, a bank and an insurance company can conduct FL to improve their predictions (e.g., whether a person can repay the loan and whether the person will buy the insurance products), while even the features can vary a lot in these organizations. Techniques in transfer learning [134], meta-learning [52], and multi-task learning [144] may be useful to combine the knowledge of various kinds of parties.

2.3.2 Manager

In the cross-device setting, the manager is usually a powerful central server. It conducts the training of the global machine learning model and manages the communication between the parties and the server. The stability and reliability of the server are quite important. Once the server fails to provide the accurate computation results, the FLS may produce a bad model. To address these potential issues, blockchain [164] may be a possible technique to offer a decentralized solution in order to increase the system reliability. For example, Kim *et al.* [85] leverage the blockchain in lieu of the central server in their system, where the blockchain enables exchanging the devices' updates and providing rewards to them.

In the cross-silo setting, since the organizations are expected to have powerful machines, the manager can also be one of the organizations who dominates the FL process. This is particularly used in the vertical FL [191], which we will introduce in Section 3.1 in detail. In a vertical FL setting by Liu *et al.* [110], the features of data are vertically



① Send the global model to the selected parties
② Update model with local data
③ Send local models to the server
④ Update the global model
① Update the local gradients
② Send the gradients to the selected party
③ Update model with local data and all gradients
④ Send the model to the other parties

(a) FedAvg (b) SimFL

Fig. 1. Federated learning frameworks.

partitioned across the parties and only one party has the labels. The party that owns the labels is naturally considered as the FL manager.

One challenge can be that it is hard to find a trusted server or party as the manager, especially in the cross-silo setting. Then, a fully-decentralized setting can be a good choice, where the parties communicate with each other directly and almost equally contribute to the global machine learning model training. These parties jointly set a FL task and deploy the FLS. Li *et al.* [95] propose a federated gradient boosting decision trees framework, where each party trains decision trees sequentially and the final model is the combination of all trees. It is challenging to design a fully-decentralized FLS with reasonable communication overhead.

2.3.3 Communication-Computation Framework

In FLSs, the computation happens on the parties and the manager, while the communication happens between the parties and the manager. Usually, the aim of the computation is for the model training and the aim of the communication is for exchanging the model parameters.

A basic and widely used framework is Federated Averaging (FedAvg) [119] proposed in 2016, as shown in Fig. 1a. In each iteration, the server first sends the current global model to the selected parties. Then, the selected parties update the global model with their local data. Next, the updated models are sent back to the server. Last, the server averages all the received local models to get a new global model. FedAvg repeats the above process until reaching the specified number of iterations. The global model of the server is the final output.

While FedAvg is a centralized FL framework, SimFL, proposed by Li *et al.* [101], represents a decentralized FL framework. In SimFL, no trusted server is needed. In each iteration, the parties first update the gradients of their local data. Then, the gradients are sent to a selected party. Next, the selected party use its local data and the gradients to update the model. Last, the model is sent to all the other parties. To ensure fairness and utilize the data from different parties, every party is selected for updating the model for about the same number of rounds. SimFL repeats a specified number of iterations and outputs the final model.

3 Taxonomy

Considering the common system abstractions and building blocks for different FLSs, we classify FLSs by six aspects:

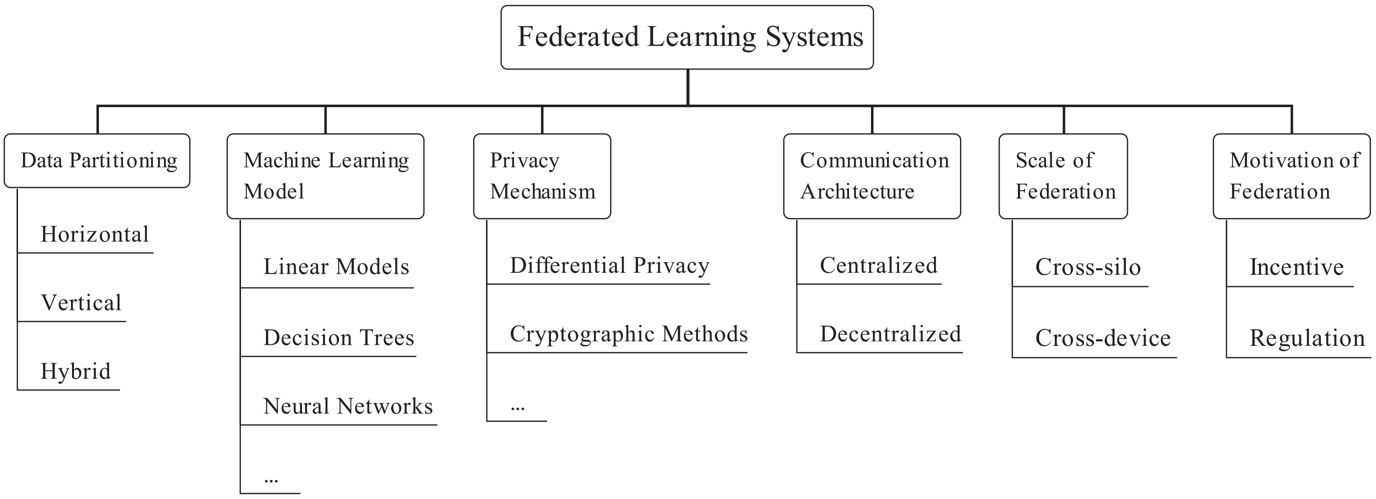


Fig. 2. Taxonomy of federated learning systems.

data partitioning, machine learning model, privacy mechanism, communication architecture, scale of federation, and motivation of federation. These aspects include common factors (e.g., data partitioning, communication architecture) in previous FLSs [89], [153] and unique consideration (e.g., machine learning model and privacy mechanism) for FLSs. Furthermore, these aspects can be used to guide the design of FLSs. Fig. 2 shows the summary of the taxonomy of FLSs.

In Table 1 of [77], they consider different characteristics to distinguish distributed learning, cross-device federated learning, and cross-silo federated learning, including setting, data distribution, communication, etc. Our taxonomy is used to distinguish different federated learning systems from a deployment view, and aspects like machine learning models and motivation of federation are not considered in [77].

3.1 Data Partitioning

Based on how data are distributed over the sample and feature spaces, FLSs can be typically categorized in horizontal, vertical, and hybrid FLSs [191].

In horizontal FL, the datasets of different parties have the same feature space but little intersection on the sample space. This is a natural data partitioning especially for the cross-device setting, where different users try to improve their model performance on the same task using FL. Also, the majority of FL studies adopt horizontal partitioning. Since the local data are in the same feature space, the parties can train the local models using their local data with the same model architecture. The global model can simply be updated by averaging all the local models. A basic and popular framework of horizontal federated learning is FedAvg, as shown in Fig. 1. Wake-word recognition [90], such as ‘Hey Siri’ and ‘OK Google’, is a typical application of horizontal partition because each user speaks the same sentence with a different voice.

In vertical FL, the datasets of different parties have the same or similar sample space but differ in the feature space. For the vertical FLS, it usually adopts *entity alignment* techniques [39], [190] to collect the overlapped samples of the parties. Then the overlapped data are used to train the machine learning model using encryption methods. Cheng *et al.* [36] propose a lossless vertical FLS to enable parties to collaboratively train gradient boosting decision trees. They use privacy-preserving entity alignment to find common users among two parties, whose gradients are used to jointly train the decision trees. Cooperation among different companies usually can be treated as a situation of vertical partition.

In many other applications, while existing FLSs mostly focus on one kind of partition, the partition of data among the parties may be a hybrid of horizontal partition and vertical partition. Let us take cancer diagnosis system as an example. A group of hospitals wants to build an FLS for cancer diagnosis but each hospital has different patients as well as different kinds of medical examination results. Transfer learning [134] is a possible solution for such scenarios. Liu *et al.* [110] propose a secure federated transfer learning system which can learn a representation among the features of parties using common instances.

3.2 Machine Learning Models

Since FL is used to solve machine learning problems, the parties usually want to train a state-of-the-art machine learning model on a specified task. There have been many efforts in developing new models or reinventing current models to the federated setting. Here, we consider the widely-used models nowadays. The most popular machine learning model now is neural network (NN), which achieves state-of-the-art results in many tasks such as image classification and word prediction [88], [163]. There are many federated learning studies based on stochastic gradient descent [22], [119], [174], which can be used to train NNs.

Another widely used model is decision tree, which is highly efficient to train and easy to interpret compared with NNs. A tree-based FLS is designed for the federated training of single or multiple decision trees (e.g., gradient boosting decision trees (GBDTs) and random forests). GBDTs are especially popular recently and it has a very good performance in many classification and regression tasks. Li *et al.* [95] and Cheng *et al.* [36] propose FLSs for GBDTs on horizontally and vertically partitioned data, respectively.

TABLE 1
Comparison Among Existing Published Studies

FL Studies	main area	data partitioning	model implementation	privacy mechanism	communication architecture	remark
FedAvg [119]	Effective Algorithms	horizontal	NN	╲	centralized	SGD-based
FedSVRG [86]	Effective Algorithms	horizontal	LM	╲	centralized	SGD-based
FedProx [100]	Effective Algorithms	horizontal	LM, NN	╲	centralized	SGD-based
SCAFFOLD [82]	Effective Algorithms	horizontal	LM, NN	╲	centralized	SGD-based
FedNova [175]	Effective Algorithms	horizontal	NN	╲	centralized	SGD-based
Per-FedAvg [49]	Effective Algorithms	horizontal	NN	╲	centralized	SGD-based
pFedMe [43]	Effective Algorithms	horizontal	LM, NN	╲	centralized	SGD-based
IAPGD, AL2SGD+ [63]	Effective Algorithms	horizontal	LM	╲	centralized	SGD-based
IFCA [57]	Effective Algorithms	horizontal	LM, NN	╲	centralized	SGD-based
Agnostic FL [122]	Effective Algorithms	horizontal	LM, NN	╲	centralized	SGD-based
FedRobust [142]	Effective Algorithms	horizontal	NN	╲	centralized	SGD-based
FedDF [106]	Effective Algorithms	horizontal	NN	╲	centralized	SGD-based
FedBCD [111]	Effective Algorithms	vertical	NN	╲	centralized	SGD-based
PNFM [196]	Effective Algorithms	horizontal	NN	╲	centralized	NN-specialized
FedMA [174]	Effective Algorithms	horizontal	NN	╲	centralized	NN-specialized
SplitNN [174]	Effective Algorithms	vertical	NN	╲	centralized	NN-specialized
Tree-based FL [200]	Effective Algorithms	horizontal	DT	DP	decentralized	DT-specialized
SimFL [95]	Effective Algorithms	horizontal	DT	hashing	decentralized	DT-specialized
FedXGB [113]	Effective Algorithms	horizontal	DT	CM	centralized	DT-specialized
FedForest [112]	Effective Algorithms	horizontal	DT	CM	centralized	DT-specialized
SecureBoost [36]	Effective Algorithms	vertical	DT	CM	centralized	DT-specialized
Ridge Regression FL [128]	Effective Algorithms	horizontal	LM	CM	centralized	LM-specialized
PPRR [34]	Effective Algorithms	horizontal	LM	CM	centralized	LM-specialized
Linear Regression FL [149]	Effective Algorithms	vertical	LM	CM	centralized	LM-specialized
Logistic Regression FL [66]	Effective Algorithms	horizontal	LM	CM	centralized	LM-specialized
Federated MTL [156]	Effective Algorithms	horizontal	LM	╲	centralized	multi-task learning
Federated Meta-Learning [31]	Effective Algorithms	horizontal	NN	╲	centralized	meta-learning
Personalized FedAvg [74]	Effective Algorithms	horizontal	NN	╲	centralized	meta-learning
LFRL [107]	Effective Algorithms	horizontal	NN	╲	centralized	reinforcement learning
FBO [41]	Effective Algorithms	horizontal	LM	╲	centralized	Bayesian optimization
Structure Updates [87]	Practicality Enhancement	horizontal	NN	╲	centralized	efficiency improvement
Multi-Objective FL [208]	Practicality Enhancement	horizontal	NN	╲	centralized	efficiency improvement
On-Device ML [72]	Practicality Enhancement	horizontal	NN	╲	centralized	efficiency improvement
Sparse Ternary Compression [151]	Practicality Enhancement	horizontal	NN	╲	centralized	efficiency improvement
DPASGD [118]	Practicality Enhancement	horizontal	NN	╲	decentralized	efficiency improvement
Client-Level DP FL [56]	Practicality Enhancement	horizontal	NN	DP	centralized	privacy guarantees
FL-LSTM [120]	Practicality Enhancement	horizontal	NN	DP	centralized	privacy guarantees
Local DP FL [18]	Practicality Enhancement	horizontal	LM, NN	DP	centralized	privacy guarantees
Secure Aggregation FL [21]	Practicality Enhancement	horizontal	NN	CM	centralized	privacy guarantees
Hybrid FL [168]	Practicality Enhancement	horizontal	LM, DT, NN	CM, DP	centralized	privacy guarantees
Backdoor FL [14, 162, 173]	Practicality Enhancement	horizontal	NN	╲	centralized	robustness and attacks
Adversarial Lens [17]	Practicality Enhancement	horizontal	NN	╲	centralized	robustness and attacks
Distributed Backdoor [187]	Practicality Enhancement	horizontal	NN	╲	centralized	robustness and attacks
Image Reconstruction [54]	Practicality Enhancement	horizontal	NN	╲	centralized	robustness and attacks
RSA [92]	Practicality Enhancement	horizontal	LM	╲	centralized	robustness and attacks
Model Poison [50]	Practicality Enhancement	horizontal	LM, NN	╲	centralized	robustness and attacks
q-FedAvg [102]	Practicality Enhancement	horizontal	LM, NN	╲	centralized	fairness
BlockFL [85]	Practicality Enhancement	horizontal	LM	╲	centralized	incentives
Reputation FL [79]	Practicality Enhancement	horizontal	LM	╲	centralized	incentives
FedCS [130]	Applications	horizontal	NN	╲	centralized	edge computing
DRL-MEC [179]	Applications	horizontal	NN	╲	centralized	edge computing
Resource-Constrained MEC [177]	Applications	horizontal	LM, NN	╲	centralized	edge computing
FedGKT [67]	Applications	horizontal	NN	╲	centralized	edge computing
FedCF [12]	Applications	horizontal	LM	╲	centralized	collaborative filter
FedMF [27]	Applications	horizontal	LM	╲	centralized	matrix factorization
FedRecSys [165]	Applications	horizontal	LM, NN	CM	centralized	recommender system
FL Keyboard [65]	Applications	horizontal	NN	╲	centralized	natural language processing
Fraud detection [204]	Applications	horizontal	NN	╲	centralized	credit card transaction
FedML [68]	Benchmarks	horizontal &vertical	LM, NN	╲	centralized &decentralized	general purpose benchmarks
FedEval [28]	Benchmarks	horizontal	NN	╲	centralized	general purpose benchmarks
OARF [70]	Benchmarks	horizontal	NN	CM,DP	centralized	general purpose benchmarks
Edge AIBench [64]	Benchmarks	horizontal	╲	╲	╲	general purpose benchmarks
PerfEval [129]	Benchmarks	horizontal	NN	╲	centralized	targeted benchmarks
FedReID [209]	Benchmarks	horizontal	NN	╲	centralized	targeted benchmarks
semi-supervised benchmark [199]	Benchmarks	horizontal	NN	╲	centralized	targeted benchmarks
non-IID benchmark [109]	Benchmarks	horizontal	╲	╲	╲	targeted benchmarks
LEAF [23]	Benchmarks	horizontal	╲	╲	centralized	datasets
Street Dataset [115]	Benchmarks	horizontal	╲	╲	╲	datasets

LM denotes Linear Models. DM denotes Decision Trees. NN denotes Neural Networks. CM denotes Cryptographic Methods. DP denotes Differential Privacy.

Besides NNs and trees, linear models (e.g., linear regression, logistic regression, SVM) are classic and easy-to-use models. There are some well developed systems for linear regression and logistic regression [66], [128]. These linear models are easy to learn compared with other complex models (e.g., NNs).

While a single machine learning model may be weak, ensemble methods [137] such as stacking and voting can be applied in the federated setting. Each party trains a local model and sends it to the server, which aggregates all the models as an ensemble. The ensemble can directly be used for prediction by max voting or be used to train a meta-model by stacking. A benefit of federated ensemble learning is that each party can train heterogeneous models as there is no averaging of model parameters. As shown in previous studies [98], [196], federated ensemble learning can also achieve a good accuracy in a single communication round.

Currently, many FL frameworks [86], [100], [119], [177] are proposed based on stochastic gradient descent, which is a typical optimization algorithm for many models including neural networks and logistic regression. However, to increase the effectiveness of FL, we may have to exploit the model architecture [174]. Since the research of FL is still at an early stage, there is still a gap for FLSs to better support the state-of-the-art models.

3.3 Privacy Mechanisms

Although the local data are not exposed in FL, the exchanged model parameters may still leak sensitive information about the data. There have been many attacks against machine learning models [53], [121], [125], [154], such as model inversion attack [53] and membership inference attack [154], which can potentially infer the raw data by accessing to the model. Moreover, there are many privacy mechanisms such as differential privacy [45] and k-anonymity [47], which provide different privacy guarantees. The characteristics of existing privacy mechanisms are summarized in the survey [171]. Here we introduce two major approaches that are adopted in the current FLSs for data protection: cryptographic methods and differential privacy.

Cryptographic methods such as homomorphic encryption [13], [26], [66], [108], [143], and secure multi-party computation (SMC) [20], [21], [30], [152] are widely used in privacy-preserving machine learning algorithms. Basically, the parties have to encrypt their messages before sending, operate on the encrypted messages, and decrypt the encrypted output to get the final result. Applying the above methods, the user privacy of FLSs can usually be well protected [81], [83], [131], [194]. For example, SMC [59] guarantees that all the parties cannot learn anything except the output, which can be used to securely aggregate the transferred gradients. However, SMC does not provide privacy guarantees for the final model, which is still vulnerable to the inference attacks and model inversion attacks [53], [154]. Also, due to the additional encryption and decryption operations, such systems suffer from the extremely high computation overhead.

Differential privacy [45], [46] guarantees that one single record does not influence much on the output of a function. Many studies adopt differential privacy [7], [29], [96], [104], [167], [203] for data privacy protection to ensure the parties cannot know whether an individual record participates in the learning or not. By injecting random noises to the data or model parameters [7], [96], [158], [186], differential privacy provides statistical privacy guarantees for individual records and protection against the inference attack on the model. Due to the noises in the learning process, such systems tend to produce less accurate models.

Note that the above methods are independent of each other, and an FLS can adopt multiple methods to enhance the privacy guarantees [60], [78], [189]. There are also other approaches to protect the user privacy. An interesting hardware-based approach is to use trusted execution environment (TEE) such as Intel SGX processors [132], [146], which can guarantee that the code and data loaded inside are protected. Such environment can be used inside the central server to increase its credibility.

Related to privacy level, the threat models also vary in FLSs [116]. The attacks can come from any stage of the process of FL, including inputs, the learning process, and the learnt model.

- *Inputs*. The malicious parties can conduct data poisoning attacks [11], [33], [91] on FL. For example, the parties can modify the labels of training samples with a specific class, so that the final model performs badly on this class.
- *Learning process*. During the learning process, the parties can perform model poisoning attacks [14], [187] to upload designed model parameters. Like data poisoning attacks, the global model can have a very low accuracy due to the poisoned local updating. Besides model poisoning attacks, the Byzantine fault [19], [25], [161] is also a common issue in distributed learning, where the parties may behave arbitrarily badly and upload random updates.
- *The learnt model*. If the learnt model is published, the inference attacks [53], [121], [125], [154] can be conducted on it. The server can infer sensitive information about the training data from the exchanged model parameters. For example, membership inference attacks [125], [154] can infer whether a specific data record is used in the training. Note that the inference attacks may also be conducted in the learning process by the FL manager, who has access to the local updates of the parties.

3.4 Communication Architecture

There are two major ways of communication in FLSs: centralized design and decentralized design. In the centralized design, the data flow is often asymmetric, which means the manager aggregates the information (e.g., local models) from parties and sends back training results [22]. The parameter updates on the global model are always done in this manager. The communication between the manager and the local parties can be synchronous [119] or asynchronous [159], [188]. In a decentralized design, the communications are performed among the parties [95], [200] and every party is able to update the global parameters directly.

Google Keyboard [65] is a case of centralized architecture. The server collects local model updates from users' devices and trains a global model, which is sent back to the users for inference, as shown in Fig. 1a. The scalability and stability are two important factors in the system design of the centralized FL.

While the centralized design is widely used in existing studies, the decentralized design is preferred at some aspects since concentrating information on one server may bring potential risks or unfairness. However, the design of the decentralized communication architecture is challenging, which should take fairness and communication overhead into consideration. There are currently three different decentralized designs: P2P [95], [200], graph [118] or blockchain [176], [202]. In a P2P design, the parties are equally privileged and treated during federated learning. An example is SimFL [95], where each party trains a tree sequentially and sends the tree to all the other parties. The communication architecture can also be modeled as a graph with the additional constrains such as latency and computation time. Marfoq *et al.* [118] propose an algorithm to find a throughput-optimal topology design. Recently, blockchain [205] is a popular decentralized platform for consideration. It can be used to store the information of parties in federated learning and ensure the transparency of federated learning [176].

3.5 Scale of Federation

The FLSs can be categorized into two typical types by the scale of federation: cross-silo FLSs and cross-device FLSs [77]. The differences between them lie on the number of parties and the amount of data stored in each party.

In cross-silo FLS, the parties are organizations or data centers. There are usually a relatively small number of parties and each of them has a relatively large amount of data as well as computational power. For example, Amazon wants to recommend items for users by training the shopping data collected from hundreds of data centers around the world. Each data center possesses a huge amount of data as well as sufficient computational resources. Another example is that federated learning can be used among medical institutions. Different hospitals can use federated learning to train a CNN for chest radiography classification while keeping their chest X-ray images locally [78]. With federated learning, the accuracy of the model can be significantly improved. One challenge that such FLS faces is how to efficiently distribute computation to data centers under the constraint of privacy models [206].

In cross-device FLS, on the contrary, the number of parties is relatively large and each party has a relatively small amount of data as well as computational power. The parties are usually mobile devices. Google Keyboard [192] is an example of cross-device FLSs. The query suggestions of Google Keyboard can be improved with the help of FL. Due to the energy consumption concern, the devices cannot be asked to conduct complex training tasks. Under this occasion, the system should be powerful enough to manage a large number of parties and deal with possible issues such as the unstable connection between the device and the server.

3.6 Motivation of Federation

In real-world applications of FL, individual parties need the motivation to get involved in the FLS. The motivation can be regulations or incentives. FL inside a company or an organization is usually motivated by regulations (e.g., FL across different departments of a company). For example, the department which has the transaction records of users can help another department to predict user credit by federated learning. In many cases, parties cannot be forced to provide their data by regulations. However, parties that choose to participate in federated learning can benefit from it, e.g., higher model accuracy. For example, hospitals can conduct federated learning to train a machine learning model for chest radiography classification [78] or COVID-19 detection [139]. Then, the hospitals can get a good model which has a higher accuracy than human experts and the model trained locally without federation. Another example is Google Keyboard [192]. While users have the choice to prevent Google Keyboard from utilizing their data, those who agree to upload input data may enjoy a higher accuracy of word prediction. Users may be willing to participate in federated learning for their convenience.

A challenging problem is how to design a fair incentive mechanism, such that the party that contributes more can also benefit more from federated learning. There have been some successful cases for incentive designs in blockchain [48], [210]. The parties inside the system can be collaborators as well as competitors. Other incentive designs like [79], [80] are proposed to attract participants with high-quality data for FL. We expect game theory models [76], [124], [150] and their equilibrium designs should be revisited under the FLSs. Even in the case of Google Keyboard, the users need to be motivated to participate this collaborative learning process.

4 Summary of Existing Studies

In this section,[1] we summarize and compare the existing studies on FLSs according to the aspects considered in Section 3.

4.1 Methodology

To discover the existing studies on FL, we search keyword "Federated Learning" in Google Scholar. Here we only consider the published studies in the computer science community.

Since the scale of federation and the motivation of federation are problem dependent, we do not compare the existing studies by these two aspects. For ease of presentation, we use "NN", "DT" and "LM" to denote neural networks, decision trees and linear models, respectively. Moreover, we use "CM" and "DP" to denote cryptographic methods and differential privacy, respectively. Note that the algorithms (e.g., federated stochastic gradient descent) in some studies can be used to learn many machine learning models (e.g.,

1. Last updated on January 19, 2022 We will periodically update this section to include the state-of-the-art and valuable FL studies. Please check out our latest version at this URL: https://arxiv.org/abs/1907.09693. Also, if you have any reference that you want to add into this survey, kindly drop Dr. Bingsheng He an email (hebs@comp.nus.edu.sg).

logistic regression and neural networks). Thus, in the "model implementation" column, we present the models implemented in the experiments of corresponding papers. Moreover, in the "main area" column, we indicate the major area that the papers study on.

4.2 Individual Studies

We summarize existing popular and state-of-the-art research work, as shown in Table 1. From Table 1, we have the following four key findings.

First, most of the existing studies consider a horizontal data partitioning. We conjecture a part of the reason is that the experimental studies and benchmarks in horizontal data partitioning are relatively ready than vertical data partitioning. However, vertical FL is also common in real world, especially between different organizations. Vertical FL can enable more collaboration between diverse parties. Thus, more efforts should be paid to vertical FL to fill the gap.

Second, most studies consider exchanging the raw model parameters without any privacy guarantees. This may not be right if more powerful attacks on machine learning models are discovered in the future. Currently, the mainstream methods to provide privacy guarantees are differential privacy and cryptographic methods (e.g., secure multi-party computation and homomorphic encryption). Differential privacy may influence the final model quality a lot. Moreover, the cryptographic methods bring much computation and communication overhead and may be the bottleneck of FLSs. We look forward to a cheap way with reasonable privacy guarantees to satisfy the regulations.

Third, the centralized design is the mainstream of current implementations. A trusted server is needed in their settings. However, it may be hard to find a trusted server especially in the cross-silo setting. One naive approach to remove the central server is that the parties share the model parameters with all the other parties and each party also maintains the same global model locally. This method bring more communication and computation cost compared with the centralized setting. More studies should be done for practical FL with the decentralized architecture.

Last, the main research directions (also the main challenge) of FL are to improve the effectiveness, efficiency, and privacy, which are also three important metrics to evaluate an FLS. Meanwhile, there are many other research topics on FL such as fairness and incentive mechanisms. Since FL is related to many research areas, we believe that FL will attract more researchers and we can see more interesting studies in the near future.

Due to the page limit, we present representative and popular studies to cover comprehensive topics in the following sections. We select the paper accordingly to the Google Scholar citations as well as a balance on the state-of-the-art. For the full description of all studies in Table 1, please refer to Section 3 of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2021.3124599.

4.2.1 Effectiveness Improvement

While some algorithms are based on SGD, the other algorithms are specially designed for one or several kinds of model architectures. Thus, we classify them into SGD-based algorithms and model specialized algorithms accordingly.

SGD-Based. FedAvg [119], as introduced in Section 2.3.3 and Fig. 1a, is now a typical and practical FL framework based on SGD. In FedAvg, each party conducts multiple training rounds with SGD on its local model. Then, the weights of the global model are updated as the mean of weights of the local models. The global model is sent back to the parties to finish a global iteration. By averaging the weights, the local parties can take multiple steps of gradient descent on their local models, so that the number of communication rounds can be reduced compared with FedSGD.

A key challenge in federated learning is the heterogeneity of local data (i.e., non-IID data) [97], which can degrade the performance of federated learning a lot [82], [100], [103]. Since the local models are updated towards their local optima, which are far from each other due to non-IID data, the averaged global model may also far from the global optima. To address the challenge, Li *et al* . [100] propose FedProx. Since too many local updates may lead the averaged model far from the global optima, FedProx introduces an additional proximal term in the local objective to limit the amount of local changes. Instead of directly limiting the size of local updates, SCAFFOLD [82] applies the variance reduction technique to correct the local updates. While FedProx and SCAFFOLD improve the local training stage of FedAvg, FedNova [175] improves the aggregation stage of FedAvg. It takes the heterogeneous local updates of each party into consideration and normalizes the local models accordingly before averaging.

There are few studies on SGD-based vertical federated learning. [111] propose the Federated Stochastic Block Coordinate Descent (FedBCD) for vertical FL. By applying coordinate descent, each party updates its local parameter for multiple rounds before communicating the intermediate results. They also provide convergence analysis for FedBCD. Hu *et al.* [71] propose FDML for vertical FL assuming all parties have the labels. Instead of exchanging the intermediate results, it aggregates the local prediction from each of the participated party.

Neural Networks. Although neural networks can be trained using the SGD optimizer, we can potentially increase the model utility if the model architecture can also be exploited. Yurochkin *et al.* [196] develop probabilistic federated neural matching (PFNM) for multilayer perceptrons by applying Bayesian nonparametric machinery [55]. They use an Beta-Bernoulli process informed matching procedure to combine the local models into a federated global model. The experiments show that their approach can outperform FedAvg on both IID and non-IID data partitioning.

Wang *et al.* [174] show how to apply PFNM to CNNs (convolutional neural networks) and LSTMs (long short-term memory networks). Moreover, they propose Federated Matched Averaging (FedMA) with a layer-wise matching scheme by exploting the model architecture. The experiments show that FedMA performs better than FedAvg and FedProx [100] on CNNs and LSTMs.

Trees. Besides neural networks, decision trees are also widely used in the academic and industry [32], [51], [84], [96]. Compared with NNs, the training and inference of trees are highly efficient. However, the tree parameters cannot be directly optimized by SGD, which means that SGD-based FL frameworks are not applicable to learn trees. We need specialized frameworks for trees. Among the tree models, the Gradient Boosting Decision Tree (GBDT) model [32] is quite popular. There are several studies on federated GBDT.

There are some studies on horizontal federated GBDTs. Zhao *et al.* [200] propose the first FLS for GBDTs. In their framework, each decision tree is trained locally without the communications between parties. The trees trained in a party are sent to the next party to continuous train a number of trees. Differential privacy is used to protect the decision trees. Li *et al.* [95] exploit similarity information in the building of federated GBDTs by using locality-sensitive hashing [42]. They utilize the data distribution of local parties by aggregating gradients of similar instances. Within a weaker privacy model compared with secure multi-party computation, their approach is effective and efficient.

Linear/Logistic Regression. Linear/logistic regression can be achieved using SGD. Here we show the studies that are not SGD-based and specially designed for linear/logistic regression.

In the horizontal FL setting, Nikolaenko *et al.* [128] propose a system for privacy-preserving ridge regression. Their approaches combine both homomorphic encryption and Yao's garbled circuit to achieve privacy requirements. An extra evaluator is needed to run the algorithm. In the vertical FL setting, Sanil *et al.* [149] present a secure regression model. They focus on the linear regression model and secret sharing is applied to ensure privacy in their solution. Hardy *et al.* [66] present a solution for two-party vertical federated logistic regression. They apply entity resolution and additively homomorphic encryption.

Others. There are many studies that combine FL with other machine learning techniques such as multi-task learning [144], meta-learning [52], and transfer learning [134].

Smith *et al.* [156] combine FL with multi-task learning [24], [198]. Their method considers the issues of high communication cost, stragglers, and fault tolerance for MTL in the federated environment. Chen *et al.* [31] adopt meta-learning in the learning process of FedAvg. Instead of training the local NNs and exchanging the model parameters, the parties adopt the Model-Agnostic Meta-Learning (MAML) [52] algorithm in the local training and exchange the gradients of MAML. Dai *et al.* [41] considers Bayesian optimization in federated learning. They propose federated Thompson sampling to address the communication efficiency and heterogeneity of the clients. Their approach can potentially be used in the parameter search in federated learning.

Another issue in FL is the package loss or party disconnection during FL process, which usually happens on mobile devices. When the number of failed messages is small, the server can simply ignore them as they have a small weight on the updating of the global model. If the party failure is significant, the server can restart from the results of the previous round [22]. We look forward to more novel solutions to deal with the disconnection issue for effectiveness improvement.

Summary. We summarize the above studies as follows.

- As the SGD-based framework has been widely studied and used, more studies focus on model specialized FL recently. We expect to achieve better model accuracy by using model specialized methods. Moreover, we encourage researchers to study on federated decision trees models. The tree models have a small model size and are easy to train compared with neural networks, which can result in a low communication and computation overhead in FL.
- The study on FL is still on a early stage. Few studies have been done on appling FL to train the state-of-the-art neural networks such as ResNeXt [117] and EfficientNet [166]. How to design an effective and practical algorithm to train a complex machine learning model is still a challenging and on-going research direction.
- While most studies focus on horizontal FL, there is still no well developed algorithm for vertical FL. However, the vertical federated setting is common in real world applications where multiple organizations are involved. We look forward to more studies on this promising area.

4.2.2 Practicality Enhancement

Communication Efficiency. While the computation of FL can be accelerated using modern hardware and techniques [93], [94], [114] in high performance computing community [181], [183], the FL studies mainly work on reducing the communication size during the FL process.

Konečný *et al.* [87] propose two ways, structured updates and sketched updates, to reduce the communication costs in FedAvg. The first approach restricts the structure of local updates and transforms it to the multiplication of two smaller matrices. Only one small matrix is sent during the learning process. The second approach uses a lossy compression method to compress the updates. Their method can reduce the communication cost by two orders of magnitude with a slight degradation in convergence speed.

Beside the communication size, the communication architecture can also be improved to increase the training efficiency. Marfoq *et al* . [118] consider the topology design for cross-silo federated learning. They propose an approach to find a throughput-optimal topology, which can significantly reduce the training time.

Privacy, Robustness and Attacks. Although the raw data is not exchanged in FL, the model parameters may leak sensitive information about the training data [125], [154]. Thus, it is important to provide privacy guarantees for the exchanged messages.

Differential privacy is a popular method to provide privacy guarantees. Geyer *et al.* [56] apply differential privacy in federated averaging from a client-level perspective. They use the Gaussian mechanism to distort the sum of updates of gradients to protect a whole client's dataset instead of a single data point.

Truex *et al.* [168] combine both secure multiparty computation and differential privacy for privacy-preserving FL. They use differential privacy to inject noises to the local updates. Then the noisy updates will be encrypted using the Paillier cryptosystem [133] before sent to the central server.

For the attacks on FL, one kind of popular attack is backdoor attack, which aims to achieve a bad global model by exchanging malicious local updates. Bagdasaryan *et al.* [14] conduct model poisoning attack on FL. The malicious parties commit the attack models to the server so that the global model may overfit with the poisoned data. The secure multi-party computation cannot prevent such attack since it aims to protect the confidentiality of the model parameters. Xie *et al.* [187] propose a distributed backdoor attack on FL. They decompose the global trigger pattern into local patterns. Each adversarial party only employs one local pattern. The experiments show that their distributed backdoor attack outperforms the central backdoor attack.

Fairness and Incentive Mechanisms. By taking fairness into consideration based on FedAvg, Li *et al.* [102] propose q-FedAvg. Specifically, they define the fairness according to the variance of the performance of the model on the parties. If such variance is smaller, then the model is more fair. Thus, they design a new objective inspired by α-fairness. Based on federated averaging, they propose q-FedAvg to solve their new objective. The major difference between q-FedAvg with FedAvg is in the formulas to update model parameters.

Kim *et al.* [85] combine the blockchain architecture with FL. On the basis of federated averaging, they use a blockchain network to exchange the devices' local model updates, which is more stable than a central server and can provide the rewards for the devices.

Summary. We summarize the above studies as follows.

- Besides effectiveness, efficiency and privacy are the other two important factors of an FLS. Compared with these three areas, there are fewer studies on fairness and incentive mechanisms. We look forward to more studies on fairness and incentive mechanisms, which can encourage the usage of FL in the real world.
- For the efficiency improvement of FLSs, the communication overhead is still the main challenge. Most studies [72], [87], [151] try to reduce the communication size of each iteration. How to reasonably set the number of communication rounds is also promising [208]. The trade-off between the computation and communication still needs to be further investigated.
- For the privacy guarantees, differential privacy and secure multi-party computation are two popular techniques. However, differential privacy may impact the model quality significantly and secure multi-party computation may be very time-consuming. It is still challenging to design a practical FLS with strong privacy guarantees. Also, the effective robust algorithms against poisoning attacks are not widely adopted yet.

4.2.3 Applications

One related area with FL is edge computing [44], [127], [140], [195], [201], where the parties are edge devices. Many studies try to integrate FL with the mobile edge systems. FL also shows promising results in recommender system [12], [27], [207], natural language processing [65], etc. Here we introduce the studies on edge computing and recommender systems.

Edge Computing. Nishio and Yonetani [130] implement federated averaging in practical mobile edge computing (MEC) frameworks. They use an operator of MEC frameworks to manage the resources of heterogeneous clients. Wang *et al.* [177] perform FL on resource-constrained MEC systems. They address the problem of how to efficiently utilize the limited computation and communication resources at the edge. Using federated averaging, they implement many machine learning algorithms including linear regression, SVM, and CNN. He *et al.* [67] also consider the limited computing resources in the edge devices. They propose FedGKT, where each device only trains a small part of a whole ResNet to reduce the computation overhead.

Recommender Systems. Ammad-ud din *et al.* [12] formulate the first federated collaborative filter method. Based on a stochastic gradient approach, the item-factor matrix is trained in a global server by aggregating the local updates. They empirically show that the federated method has almost no accuracy loss compared with the centralized method. Tan *et al.* [165] build a federated recommender system (FedRecSys) based on FATE. FedRecSys has implemented popular recommendation algorithms with SMC protocols. The algorithms include matrix factorization, singular value decomposition, factorization machine, and deep learning.

Summary. We summarize the above studies as follows.

- Edge computing naturally fits the cross-device federated setting. A nontrivial issue of applying FL to edge computing is how to effectively utilize and manage the edge resources. The usage of FL can bring benefits to the users, especially for improving the mobile device services.
- FL can solve many traditional machine learning tasks such as image classification and work prediction. Due to the regulations and "data islands", the federated setting may be a common setting in the next years. With the fast development of FL, we believe that there will be more applications in computer vision, natural language processing, and healthcare.

4.2.4 Benchmark

Benchmark is important for directing the development of FLSs. Multiple benchmark-related works have been conducted recently, and several benchmark frameworks are available online. We categorize them into three types: 1) *General purpose benchmark systems* aim at comprehensively evaluate FLSs and give a detailed characterization of different aspects of FLSs; 2) *Targeted benchmarks* aim at one or more aspects that concentrated in a small domain and tries to optimize the performance of the system in that domain; 3) *Dataset benchmarks* aim at providing dedicated datasets for federated learning.

General Purpose Benchmark Systems. FedML [68] is a research library that provides both frameworks for federated learning and benchmark functionalities. As a benchmark, it provides comprehensive baseline implementations for multiple ML models and FL algorithms, including FedAvg, FedNAS, Vertical FL, and split learning. Moreover, it supports three computing paradigms, namely distributed training, mobile on-device training, and standalone simulation. Although some of its experiment results are currently still at a preliminary stage, it is one of the most comprehensive benchmark frameworks concerning its functionalities.

OARF [70] provides a set of utilities and reference implementations for FL benchmarks. It features the measurement of different components in FLSs, including FL algorithms, encryption mechanisms, privacy mechanisms, and communication methods. In addition, it also features realistic partitioning of datasets, which utilizes public datasets collected from different sources to reflect real-world data distributions.

Targeted Benchmarks. Nilsson *et al.* [129] propose a method utilizing correlated t-test to compare between different types of federated learning algorithms while bypassing the influence of data distributions. Three FL algorithms, FedAvg, FedSVRG [87] and CO-OP [180] are compared in both IID and non-IID setup in their work, and the result shows that FedAvg achieves the highest accuracy among the three algorithms regardless of data partitioning.

Zhang *et al.* [199] present a benchmark targeting at semi-supervised federated learning setting, where users only have unlabelled data, and the server only has a small amount of labelled data, and explore the relation between final model accuracy and multiple metrics, including the distribution of the data, the algorithm and communication settings, and the number of clients. Utilizing the experiment results, their semi-supervised learning improved method achieves better generalization performance.

Datasets. LEAF [23] is one of the earliest dataset proposals for federated learning. It contains six datasets covering different domains, including image classification, sentiment analysis, and next-character prediction. A set of utilities is provided to divide datasets into different parties in an IID or non-IID way. For each dataset, a reference implementation is also provided to demonstrate the usage of that dataset in the training process.

Summary. We summarize the above studies as follows.

- Benchmarks serve an important role in the development of federated learning. Through different types of benchmarks, we can quantitatively characterize the different components and aspects of federated learning. Benchmarks regarding the security and privacy issues in federated learning are still at an early stage and require further development.
- Currently no comprehensive enough benchmark system has been implemented to cover all algorithms or application types in FLSs. Even the most comprehensive benchmark systems lack supports for certain algorithms and evaluation metrics for each level of the system. Further development of comprehensive benchmark systems requires the support of extensive FL frameworks.
- Most benchmark researches are using datasets which are split from a single dataset, and there is no consensus on what type of splitting method should be used. Similarly, regarding the non-IID problem, there is no consensus on the metric of non-IID-ness. Using realistic partitioning method, as proposed in FedML [68] and OARF [70] may mitigate this issue, but for federated learning at a large-scale, realistic partitioning is not suitable due to the difficulty of collecting data from different sources.

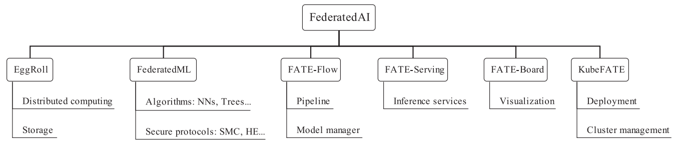


Fig. 3. The FATE system structure.

4.3 Open Source Systems

In this section, we introduce five open source FLSs: Federated AI Technology Enabler (FATE),[2] Google TensorFlow Federated (TFF),[3] OpenMined PySyft,[4] Baidu PaddleFL,[5] and FedML.[6]

4.3.1 *FATE*

FATE is an industrial level FL framework developed by WeBank, which aims to provide FL services between different organizations. FATE is based on Python and can be installed on Linux or Mac. It has attracted about 3.2k stars and 900 forks on GitHub. The overall structure of FATE is shown in Fig. 3. It has six major modules: EggRoll, FederatedML, FATE-Flow, FATE-Serving, FATE-Board, and KubeFATE. EggRoll manages the distributed computing and storage. It provides computing and storage AIPs for the other modules. FederatedML includes the federated algorithms and secure protocols. Currently, it supports training many kinds of machine learning models under both horizontal and vertical federated setting, including NNs, GBDTs, and logistic regression. FATE assumes that the parties are honest-but-curious. Thus, it uses secure multi-party computation and homomorphic encryption to protect the communicated messages. However, it does not support differential privacy to protect the final model. FATE-Flow is a platform for the users to define their pipeline of the FL process. The pipeline can include the data preprocessing, federated training, federated evaluation, model management, and model publishing. FATE-Serving provides inference services for the users. It supports loading the FL models and conducting online inference on them. FATE-Board is a visualization tool for FATE. It provides a visual way to track the job execution and model performance. Last, KubeFATE helps deploy FATE on clusters by using Docker or Kubernetes. It provides customized deployment and cluster management services. In general, FATE is a powerful and easy-to-use FLS. Users can simply set the parameters to run a FL algorithm. Moreover, FATE provides detailed documents on its deployment and usage. However, since FATE provides algorithm-level interfaces, practitioners have to modify the source code of FATE to implement their own federated algorithms. This is not easy for non-expert users.

2. https://github.com/FederatedAI/FATE
3. https://github.com/tensorflow/federated
4. https://github.com/OpenMined/PySyft
5. https://github.com/PaddlePaddle/PaddleFL
6. https://github.com/FedML-AI/FedML

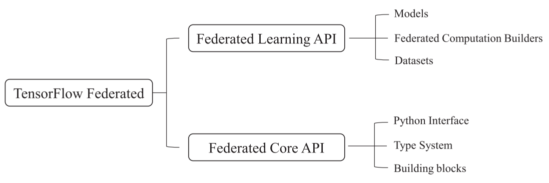


Fig. 4. The TFF system structure.

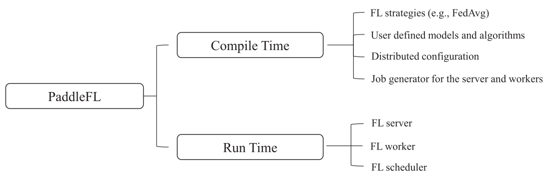

PaddleFL
Compile Time
FL strategies (e.g., FedAvg)
User defined models and algorithms
Distributed configuration
Job generator for the server and workers
Run Time
FL server
FL worker
FL scheduler

Fig. 5. The PaddleFL system structure.

4.3.2 TFF

TFF, developed by Google, provides the building blocks for FL based on TensorFlow. It has attracted about 1.5k stars and 380 forks on GitHub. TFF provides a Python package which can be easily installed and imported. As shown in Fig. 4, it provides two APIs of different layers: FL API and Federated Core (FC) API. FL API offers high-level interfaces. It includes three key parts, which are models, federated computation builders, and datasets. FL API allows users to define the models or simply load the Keras [61] model. The federated computation builders include the typical federated averaging algorithm. Also, FL API provides simulated federated datasets and functions to access and enumerate the local datasets for FL. Besides high-level interfaces, FC API also includes lower-level interfaces as the foundation of the FL process. Developers can implement their functions and interfaces inside the federated core. Finally, FC provides the building blocks for FL. It support multiple federated operators such as federated sum, federated reduce, and federated broadcast. Developers can define their own operators to implement the FL algorithm. Overall, TFF is a lightweight system for developers to design and implement new FL algorithms. Currently, TFF does not consider consider any adversaries during FL training. It does not provide privacy mechanisms. TFF can only deploy on a single machine now, where the federated setting is implemented by simulation.

4.3.3 PySyft

PySyft, first proposed by Ryffel *et al.* [145] and developed by OpenMined, is a python library that provides interfaces for developers to implement their training algorithm. It has attracted about 7.3k stars and 1.7k forks on GitHub. While TFF is based on TensorFlow, PySyft can work well with both PyTorch and TensorFlow. PySyft provides multiple optional privacy mechanisms including secure multi-party computation and differential privacy. Thus, it can support running on honest-but-curious parties. Moreover, it can be deployed on a single machine or multiple machines, where the communication between different clients is through the websocket API [155]. However, while PySyft provides a set of tutorials, there is no detailed document on its interfaces and system architecture.

4.3.4 PaddleFL

PaddleFL is a FLS based on PaddlePaddle,[7] which is a deep learning platform developed by Baidu. It is implemented on C++ and Python. It has attracted about 260 stars and 60 forks on GitHub. Like PySyft, PaddleFL supports both differential privacy and secure multi-party computation and can work on honest-but-curious parties. The system structure of PaddleFL is shown in Fig. 5. In the compile time, there are four components including FL strategies, user defined models and algorithms, distributed training configuration, and FL job generator. The FL strategies include the horizontal FL algorithms such as FedAvg. Vertical FL algorithms will be integrated in the future. Besides the provided FL strategies, users can also define their own models and training algorithms. The distributed training configuration defines the training node information in the distributed setting. FL job generator generates the jobs for federated server and workers. In the run time, there are three components including FL server, FL worker, and FL scheduler. The server and worker are the manager and parties in FL, respectively. The scheduler selects the workers that participate in the training in each round. Currently, the development of PaddleFL is still in a early stage and the documents and examples are not clear enough.

4.3.5 FedML

FedML provides both a framework for federated learning and a platform for FL benchmark. It is developed by a team from University of Southern California [68] based on PyTorch. FedML has attracted about 660 stars and 180 forks on GitHub. As an FL framework, It's core structure is divided into two levels, as shown in Fig. 6. In the low-level FedML-core, training engine and distributed communication infrastructures are implemented.

The high-level FedML-API is built on top of them and provides training models, datasets, and FL algorithms. Reference application/benchmark implementations are further built on top of the FedML-API. While most algorithms implemented on FedML does not consider any adversaries, it supports applying differential privacy when aggregating the messages from the parties. FedML supports three computing paradigms, namely standalone simulation, distributed computing and on-device training, which provides a simulation environment for a broad spectrum of hardware requirements. Reference implementations for all supported FL algorithms are provided. Although there are still gaps between

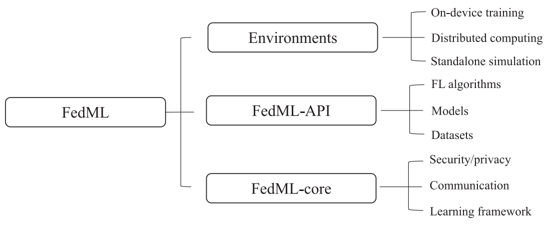


Fig. 6. The FedML system structure.

7. https://github.com/PaddlePaddle/Paddle

TABLE 2
The Comparison Among Some Existing FLSs

Supported features		FATE 1.5.0	TFF 0.17.0	PySyft 0.3.0	PaddleFL 1.1.0	FedML
Operation systems	Mac	✓	✓	✓	✓	✓
	Linux	✓	✓	✓	✓	✓
	Windows			✓	✓	✓
	iOS					✓
	Android					✓
Data partitioning	horizontal	✓	✓	✓	✓	✓
	vertical	✓			✓	✓
Models	NN	✓	✓	✓	✓	✓
	DT	✓				
	LM	✓	✓	✓	✓	✓
Privacy Mechanisms	DP		✓	✓	✓	✓
	CM	✓		✓	✓	
Communication	simulated	✓	✓	✓	✓	✓
	distributed	✓		✓	✓	✓
Hardwares	CPUs	✓	✓	✓	✓	✓
	GPUs		✓	✓		✓

The notations used in this table are the same as Table 1. The cell is left empty if the system does not support the corresponding feature. There is no release version for FedML.

some of the experiment results and the optimal results, they provide useful information for further development.

4.3.6 Others

There are other closed source federated learning systems. NVIDIA Clara[8] has enabled FL. It adopts a centralized architecture and encrypted communication channel. The targeted users of Clara FL is hospitals and medical institutions. Ping An Technology aims to build a federated learning system named Hive [2], which targets at the financial industries. While Clara FL provides APIs and documents, we cannot find the official documents of Hive.

4.3.7 Summary

Overall, FATE, PaddleFL, and FedML try to provide algorithm-level APIs for users to use directly, while TFF and PySyft try to provide more detailed building blocks so that the developers can easily implement their FL process. Table 2 shows the comparison between the open-source systems. In the algorithm level, FATE is the most comprehensive system that supports many machine learning models under both horizontal and vertical settings. TFF and PySyft only implement FedAvg, which is a basic framework in FL as shown in Section 4.2. PaddleFL supports several horizontal FL algorithms currently on NNs and logistic regression. FedML integrates several state-of-the-art FL algorithms such as FedOpt [141] and FedNova [175]. Compared with FATE, TFF, and FedML, PySyft and PaddleFL provide more privacy mechanisms. PySyft covers all the listed features that TFF supports, while TFF is based on TensorFlow and PySyft works better on PyTorch. Based on the popularity on GitHub, PySyft is currently the most impactful federated learning system in the machine learning community.

8. https://developer.nvidia.com/clara

5 System Design

Fig. 7 shows the factors that need to be considered in the design of an FLS. Here effectiveness, efficiency, and privacy are three important metrics of FLSs, which are also main research directions of federated learning. Inspired by federated database [153], we also consider autonomy, which is necessary to make FLSs practical. Next, we explain these factors in detail.

5.1 Effectiveness

The core of an FLS is an (multiple) effective algorithm (algorithms). To determine the algorithm to be implemented from lots of existing studies as shown in Table 1, we should first check the data partitioning of the parties. If the parties have the same features but different samples, one can use FedAvg [119] for NNs and SimFL [95] for trees. If the parties have the same sample space but different features, one can use FedBCD [111] for NNs and SecureBoost [36] for trees.

5.2 Privacy

An important requirement of FLSs is to protect the user privacy. Here we analyze the reliability of the manager. If the

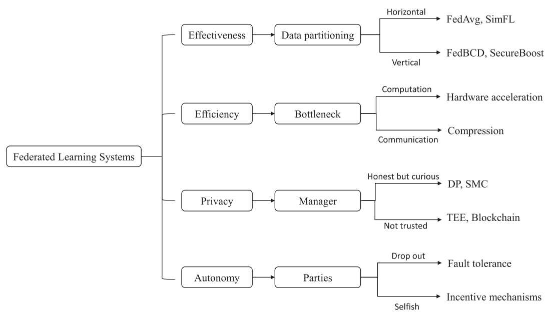


Fig. 7. The design factors of FLSs.

manager is honest and not curious, then we do not need to adopt any additional technique, since the FL framework ensures that the raw data is not exchanged. If the manager is honest but curious, then we have to take possible inference attacks into consideration. The model parameters may also expose sensitive information about the training data. One can adopt differential privacy [38], [56], [120] to inject random noises into the parameters or use SMC [20], [21], [66] to exchanged encrypted parameters. If the manager cannot be trusted at all, then we can use trusted execution environments [35] to execute the code in the manager. Blockchain is also an option to play the role as a manager [85].

5.3 Efficiency

Efficiency is an important factor in the success of many existing systems such as XGBoost [32] and ThunderSVM [182]. Since federated learning involves multi-rounds training and communication, the computation and communication costs may be large, which increases the threshold of usage of FLSs. To increase the efficiency, the most effective way is to deal with the bottleneck. If the bottleneck lies in the computation, we can use powerful hardware such as GPUs [40] and TPUs [75]. If the bottleneck lies in the communication, the compression techniques [16], [87], [151] can be applied to reduce the communication size.

5.4 Autonomy

Like federated databases [153], a practical FLS has to consider the autonomy of the parties. The parties may drop out (e.g., network failure) during the FL process, especially in the cross-device setting where the scale is large and the parties are unreliable [77]. Thus, the FLS should be robust and stable, which can tolerate the failure of parties or reduce the number of failure cases. Google has developed a practical FLS [22]. In their system, they monitor devices' health statistics to avoid wasting devices' battery or bandwidth. Also, the system will complete the current round or restart from the results of the previously committed round if there are failures. Zhang *et al.* [197] propose a blockchain-based approach to detect the device disconnection. Robust secure aggregation [15] is applicable to protect the communicated message in case of party drop out. Besides the disconnection issues, the parties may be selfish and are not willing to share the model with good quality. Incentive mechanisms [79], [80] can encourage the participation of the parties and improve the final model quality.

5.5 The Design Reference

Based on our taxonomy shown in Section 3 and the design factors shown in Fig. 7, we derive a simple design reference for developing an FLS.

The first step is to identity the participated entities and the task, which significantly influence the system design. The participated entities determines the communication architecture, the data partitioning and the scale of federation. The task determines the suitable machine learning models to train. Then, we can choose or design a suitable FL algorithm according to the above attributes and Table 1. After fixing the FL algorithm, to satisfy the privacy requirements, we may determine the privacy mechanisms to protect the communicated messages. DP is preferred if efficiency is more important than model performance compared with SMC. Last, incentive mechanism can be considered to enhance the system. Existing systems [22], [68] usually do not support incentive mechanisms. However, incentive mechanisms can encourage the parties to participate and contribute in the system and make the system more attractive. Shapley value [172], [178] is a fair approach that can be considered.

For real-world applications of federated learning systems, please refer to Section 5 of the supplementary material, available online.

5.6 Evaluation

The evaluation of FLSs is very challenging. According to our studied system factors, it has to cover the following aspects: (1) model performance, (2) system security, (3) system efficiency, and (4) system robustness.

For the evaluation of the model, there are two different settings. One is to evaluate the performance (e.g., prediction accuracy) of the final global model on a global dataset. The other one is to evaluate the performance of the final local models on the corresponding non-IID local datasets. The evaluation setting depends on the objective of FL, i.e., learn a global model or learn personalized local models.

While theoretical security/privacy guarantee is a good evaluation metric for system security, another way is to conduct membership inference attacks [154] or model inversion attacks [53] to test the system security. These attacks can be conducted in two ways: (1) white-box attack: the attacker has access to all the exchanged models during the FL process. (2) black-box attack: the attacker only has access to the final output model. The attack success ratio can be an evaluation metric for the system security.

The efficiency of the system includes two parts: computation efficiency and communication efficiency. An intuitive metric is the training time, including the computation and communication time. Note that FL is usually a multi-round process. Thus, for a fair comparison, one approach is to use time per round as a metric. Another approach is to record the time or round to achieve the same target performance [82], [99].

It is challenging to quantifying the robustness of an FLS. Inspired by robust aggregation in Byzantine tolerant distributed learning [19], [157], where the number of tolerated Byzantine adversaries is used to evaluate the robustness, we can use the maximum number of disconnected parties that can tolerate during the FL process as the metric to evaluate the robustness of an FLS.

6 Vision

In this section, we show interesting directions to work on in the future. Although some directions are already covered in existing studies introduced in Section 4, we believe they are important and provide more insights on them.

6.1 Heterogeneity

The heterogeneity of the parties is an important characteristic in FLSs. Basically, the parties can differ in the accessibility, privacy requirements, contribution to the federation, and reliability. Thus, it is important to consider such practical issues in FLSs.

Dynamic Scheduling. Due to the instability of the parties, the number of parties may not be fixed during the learning process. However, the number of parties is fixed in many existing studies and they do not consider the situations where there are entries of new parties or departures of the current parties. The system should support dynamic scheduling and have the ability to adjust its strategy when there is a change in the number of parties. There are some studies addressing this issue. For example, Google's system [22] can tolerate the drop-out of the devices. Also, the emergence of blockchain [205] can be an ideal and transparent platform for multi-party learning. However, to the best of our knowledge, there is no work that study a increasing number of parties during FL. In such a case, more attention may be paid to the later parties, as the current global model may have been welled trained on existing parties.

Diverse Privacy Restrictions. Little work has considered the privacy heterogeneity of FLSs, where the parties have different privacy requirements. The existing systems adopt techniques to protect the model parameters or gradients for all the parties on the same level. However, the privacy restrictions of the parties usually differ in reality. It would be interesting to design an FLS which treats the parties differently according to their privacy restrictions. The learned model should have a better performance if we can maximize the utilization of data of each party while not violating their privacy restrictions. The heterogeneous differential privacy [8] may be useful in such settings, where users have different privacy attitudes and expectations.

Intelligent Benefits. Intuitively, one party should gain more from the FLS if it contributes more information. Existing incentive mechanisms are mostly based on Shapley values [172], [178], the computation overhead is a major concern. A computation-efficient and fair incentive mechanism needs to be developed.

6.2 System Development

To boost the development of FLSs, besides the detailed algorithm design, we need to study from a high-level view.

System Architecture. Like the parameter server [69] in deep learning which controls the parameter synchronization, some common system architectures are needed to be investigated for FL. Although FedAvg is a widely used framework, the applicable scenarios are still limited. For example, while unsupervised learning [99], [100], [119] still adopt model averaging as the model aggregation method, which cannot work if the parties want to train heterogeneous models. We want a general system architecture, which provides many aggregation methods and learning algorithms for different settings.

Model Market. Model market [169] is a promising platform for model storing, sharing, and selling. An interesting idea is to use the model market for federated learning. The party can buy the models to conduct model aggregation locally. Moreover, it can contribute its models to the market with additional information such as the target task. Such a design introduces more flexibility to the federation and is more acceptable for the organizations, since the FL just like several transactions. A well evaluation of the models is important in such systems. The incentive mechanisms may be helpful [79], [80], [185].

Benchmark. As more FLSs are being developed, a benchmark with representative data sets and workloads is quite important to evaluate the existing systems and direct future development. Although there have been quite a few benchmarks [23], [68], [70], no benchmark has been widely used in the experiments of federated learning studies. We need a robust benchmark which has representative datasets and strict privacy evaluation. Also, comprehensive evaluation metric including model performance, system efficiency, system security, and system robustness is often ignored in existing benchmarks. The evaluation of model performance on non-IID datasets and system security under data pollution needs more investigation.

Data Life Cycles. Learning is simply one aspects of a federated system. A data life cycle [138] consists of multiple stages including data creation, storage, use, share, archive and destroy. For the data security and privacy of the entire application, we need to invent new data life cycles under FL context. Although data sharing is clearly one of the focused stage, the design of FLSs also affects other stages. For example, data creation may help to prepare the data and features that are suitable for FL.

6.3 FL in Domains

Internet-of-Thing. Security and privacy issues have been a hot research area in fog computing and edge computing, due to the increasing deployment of Internet-of-thing applications. For more details, readers can refer to some recent surveys [123], [160], [193]. FL can be one potential approach in addressing the data privacy issues, while still offering reasonably good machine learning models [105], [126]. The additional key challenges come from the computation and energy constraints. The mechanisms of privacy and security introduces runtime overhead. For example, Jiang *et al.* [73] apply independent Gaussian random projection to improve the data privacy, and then the training of a deep network can be too costly. The authors need to develop a new resource scheduling algorithm to move the workload to the nodes with more computation power. Similar issues happen in other environments such as vehicle-to-vehicle networks [147].

Regulations. While FL enables collaborative learning without exposing the raw data, it is still not clear how FL comply with the existing regulations. For example, GDPR proposes limitations on data transfer. Since the model and gradients are actually not safe enough, is such limitation still apply to the model or gradients? Also, the "right to explainability" is hard to execute since the global model is an averaging of the local models. The explainability of the FL models is an open problem [62], [148]. Moreover, if a user wants to delete its data, should the global model be retrained without the data [58]? There is still a gap between the FL techniques and the regulations in reality. We may expect the cooperation between the computer science community and the law community.

7 Conclusion

Many efforts have been devoted to developing federated learning systems (FLSs). A complete overview and summary for existing FLSs is important and meaningful. Inspired by the previous federated systems, we have shown that heterogeneity and autonomy are two important factors in the design of practical FLSs. Moreover, with six different aspects, we provide a comprehensive categorization for FLSs. Based on these aspects, we also present the comparison on features and designs among existing FLSs. More importantly, we have pointed out a number of opportunities, ranging from more benchmarks to integration of emerging platforms such as blockchain. FLSs will be an exciting research direction, which calls for the effort from machine learning, system and data privacy communities.

Acknowledgments

Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore. Qinbin is also in part supported by a Google PhD Fellowship.

References

[1] California consumer privacy act home page, 2020. [Online]. Available: https://www.caprivacy.org/

[2] 2021. [Online]. Available: https://www.intel.com/content/www/us/en/customer-spotlight/stories/ping-an-sgx-customer-story.html

[3] Uber settles data breach investigation for 148 million, 2018. [Online]. Available: https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html

[4] Google is fined 57 million under europe's data privacy law, 2019. [Online]. Available: https://www.nytimes.com/2019/01/21/technology/google-europe-gdpr-fine.html

[5] 2019 is a 'fine' year: Pdpc has fined s'pore firms a record 1.29m for data breaches, 2019. [Online]. Available: https://vulcanpost.com/676006/pdpc-data-breach-singapore-2019/

[6] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Oper. Syst. Des. Implementation*, 2016, pp. 265–283.

[7] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[8] M. Alaggan, S. Gambs, and A.-M. Kermarrec, "Heterogeneous differential privacy," *J. Priv. Confidentiality*, vol. 7, no. 2, pp. 127–158, 2016, doi: 10.29012/jpc.v7i2.652.

[9] J. P. Albrecht, "How the GDPR will change the world," *Eur. Data Protection Law Rev.*, vol. 2, pp. 287–289, 2016.

[10] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 14 0699–14 0725, 2020.

[11] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1452–1458.

[12] M. Ammad-ud din *et al.*, "Federated collaborative filtering for privacy-preserving personalized recommendation system," 2019, *arXiv:1901.09888*.

[13] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[15] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1253–1269.

[16] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," *Proc. Mach. Learn. Res.*, vol. 80, pp. 559–568, 2018. [Online]. Available: http://proceedings.mlr.press/v80/bernstein18a.html

[17] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[18] A. Bhowmick, J. C. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.

[19] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[20] K. Bonawitz *et al.*, "Practical secure aggregation for federated learning on user-held data," 2016, *arXiv:1611.04482*.

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

[22] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, 2019. [Online]. Available: https://proceedings.mlsys.org/book/271.pdf

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

[24] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[25] M. Castro *et al.*, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Oper. Syst. Des. Implementation*, 1999, pp. 173–186.

[26] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptol. ePrint Arc.*, vol. 2017, 2017, Art. no. 35.

[27] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intell. Syst.*, vol. 36, no. 5, pp. 11–20, 2021, doi: 10.1109/MIS.2020.3014880.

[28] D. Chai, L. Wang, K. Chen, and Q. Yang, "FedEval: A benchmark system with a comprehensive evaluation model for federated learning," 2020, *arXiv:2011.09655*.

[29] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, 2011.

[30] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. Cryptol.*, vol. 1, pp. 65–75, 1988.

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

[32] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

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

[34] Y.-R. Chen, A. Rezapour, and W.-G. Tzeng, "Privacy-preserving ridge regression on distributed data," *Inf. Sci.*, vol. 451/452, pp. 34–49, 2018.

[35] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, 2020.

[36] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: A lossless federated learning framework," *IEEE Intell. Syst.*, vol. 36, no. 6, pp. 87–98, 2021, doi: 10.1109/MIS.2021.3082561.

[37] W. B. Chik, "The singapore personal data protection act and an assessment of future trends in data privacy reform," *Comput. Law Secur. Rev.*, vol. 29, pp. 554–575, 2013.

[38] O. Choudhury *et al.*, "Differential privacy-enabled federated learning for sensitive health data," 2019, *arXiv:1910.02578*.

[39] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Germany: Springer Science & Business Media, 2012.

[40] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing With GPUs*. London, U.K.: Newnes, 2012.

[41] Z. Dai, K. H. Low, and P. Jaillet, "Federated Bayesian optimization via thompson sampling," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 9687–9699.

[42] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, 2004, pp. 253–262.

[43] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f4f1f13c8289ac1b1ee0ff176b56fc60-Abstract.html

[44] M. Duan *et al.*, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proc. 37th IEEE Int. Conf. Comput. Des.*, 2019, pp. 246–254, doi: 10.1109/ICCD46524.2019.00038.

[45] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[46] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, pp. 211–407, 2014.

[47] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *J. Amer. Med. Informat. Assoc.*, vol. 15, pp. 627–637, 2008.

[48] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse , "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Conf. Netw. Syst. Des. Implementation*, 2016, pp. 45–59.

[49] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 3557–3568.

[50] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Conf. Secur. Symp.*, 2020, pp. 1623–1640.

[51] J. Feng, Y. Yu, and Z.-H. Zhou, "Multi-layered gradient boosting decision trees," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3555–3565.

[52] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[53] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[54] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–How easy is it to break privacy in federated learning?" in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html

[55] S. J. Gershman and D. M. Blei, "A tutorial on Bayesian nonparametric models," *J. Math. Psychol.*, vol. 56, pp. 1–12, 2012.

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

[57] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/e32cc80bf07915058ce90722ee17bb71-Abstract.html

[58] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, "Making ai forget you: Data deletion in machine learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 316.

[59] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary Version*, Citeseer, vol. 78, 1998.

[60] S. Goryczka and L. Xiong, "A comprehensive comparison of multiparty secure additions with differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 5, pp. 463–477, Sep./Oct. 2017.

[61] A. Gulli and S. Pal, *Deep Learn. With Keras*. Birmingham, U.K.: Packt Publishing Ltd, 2017.

[62] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.-Z. Yang, "Explainable artificial intelligence (XAI)," *Sci. Robotics*, vol. 4, no. 37, 2019, doi: 10.1126/scirobotics.aay7120.

[63] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik, "Lower bounds and optimal algorithms for personalized federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/187acf7982f3c169b3075132380986e4-Abstract.html

[64] T. Hao *et al.*, "Edge AiBench: Towards comprehensive end-to-end edge computing benchmarking," *Benchmarking Measuring Optimizing - 1st Bench Council Int. Symp.*, vol. 11459, pp. 23–30, 2019, doi: 10.1007/978-3-030-32813-9_3.

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

[66] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*.

[67] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 14068–14080.

[68] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," 2020, *arXiv:2007.13518*.

[69] Q. Ho *et al.*, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1223–1231.

[70] S. Hu, Y. Li, X. Liu, Q. Li, Z. Wu, and B. He, "The OARF benchmark suite: Characterization and implications for federated learning systems," 2020, *arXiv:2006.07856*.

[71] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2232–2240.

[72] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," 2018, *arXiv:1811.11479*.

[73] L. Jiang, R. Tan, X. Lou, and G. Lin, "On lightweight privacy-preserving collaborative learning for Internet-of-Things objects," in *Proc. Int. Conf. Internet Things Des. Implementation*, 2019, pp. 70–81.

[74] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.

[75] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architecture*, 2017, pp. 1–12.

[76] R. Jurca and B. Faltings, "An incentive compatible reputation mechanism," in *Proc. IEEE Int. Conf. E-Commerce*, 2003, pp. 285–292.

[77] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021, doi: 10.1561/2200000083.

[78] G. Kaissis *et al.*, "End-to-end privacy preserving deep learning on multi-institutional medical imaging," *Nature Mach. Intell.*, vol. 3, pp. 473–484, 2021.

[79] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[80] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *Proc. IEEE VTS Asia Pacific Wireless Commun. Symp.*, 2019, pp. 1–5, doi: 10.1109/VTS-APWCS.2019.8851649.

[81] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1026–1037, Sep. 2004.

[82] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[83] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *J. Official Statist.*, vol. 25, pp. 125–138, 2009.

[84] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.

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

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

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

[88] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[89] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," *Cloud Comput.*, vol. 2011, pp. 32–38, 2011.

[90] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 6341–6345.

[91] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1893–1901.

[92] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, Art. no. 190.

[93] P. Li, Y. Luo, N. Zhang, and Y. Cao, "HeteroSpark: A heterogeneous CPU/GPU spark platform for machine learning algorithms," in *Proc. IEEE Int. Conf. Netw. Archit. Storage*, 2015, pp. 347–348.

[94] Q. Li, Z. Wen, and B. He, "Adaptive kernel value caching for SVM training," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2376–2386, Jul. 2020.

[95] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 4642–4649.

[96] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 784–791.

[97] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021, *arXiv:2102.02079*.

[98] Q. Li, B. He, and D. Song, "Practical one-shot federated learning for cross-silo setting," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 1484–1490.

[99] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10708–10717.

[100] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2018. [Online]. Available: https://proceedings.mlsys.org/book/316.pdf

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

[102] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. 8th Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ByexElSYDr

[103] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. 8th Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[104] H. A. Lianto, Y. Zhao, and J. Zhao, "Attacks to federated learning: Responsive web user interface to recover training data from user gradients," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, 2020, pp. 901–903.

[105] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[106] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/18df51b97ccd68128e994804f3eccc87-Abstract.html

[107] B. Liu, L. Wang, M. Liu, and C. Xu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4555–4562, 2019, doi: 10.1109/LRA.2019.2931179.

[108] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 619–631.

[109] L. Liu, F. Zhang, J. Xiao, and C. Wu, "Evaluation framework for large-scale federated learning," 2020, *arXiv:2003.01575*.

[110] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," 2018, *arXiv:1812.03337*.

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

[112] Y. Liu, Y. Liu, Z. Liu, J. Zhang, C. Meng, and Y. Zheng, "Federated forest," 2019, *arXiv:1905.10053*.

[113] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. Deng, "Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing," 2019, *arXiv:1907.10218*.

[114] N. Lopes and B. Ribeiro, "GPUMLib: An efficient open-source GPU machine learning library," *Int. J. Comput. Inf. Syst. Ind. Manage. Appl.*, vol. 3, pp. 355–362, 2011.

[115] J. Luo *et al.*, "Real-world image datasets for federated learning," 2019, *arXiv:1910.11089*.

[116] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv:2003.02133*.

[117] D. Mahajan *et al.*, "Exploring the limits of weakly supervised pretraining," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 185–201.

[118] O. Marfoq, C. Xu, G. Neglia, and R. Vidal, "Throughput-optimal topology design for cross-silo federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/e29b722e35040b88678e25a1ec032a21-Abstract.html

[119] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, pp. 1273–1282, 2017. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[120] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. 6th Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b

[121] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 691–706.

[122] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, pp. 4615–4625, 2019. [Online]. Available: http://proceedings.mlr.press/v97/mohri19a.html

[123] M. Mukherjee *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[124] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proc. 1st ACM Conf. Electron. Commerce*, 1999, pp. 129–139.

[125] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[126] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏot: A federated self-learning anomaly detection system for IoT," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 756–767, doi: 10.1109/ICDCS.2019.00080.

[127] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, 2020.

[128] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 334–348.

[129] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proc. 2nd Workshop Distrib. Infrastructures Deep Learn.*, 2018, pp. 1–8.

[130] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[131] R. Nock *et al.*, "Entity resolution and federated learning get a federated resolution," 2018, *arXiv:1803.04035*.

[132] O. Ohrimenko *et al.*, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Conf. Secur. Symp.*, 2016, pp. 619–636.

[133] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[134] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[135] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–4.

[136] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 721.

[137] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*. Berlin, Germany: Springer, 2012.

[138] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," *ACM SIGMOD Rec.*, vol. 47, no. 2, pp. 17–28, 2018.

[139] A. Qayyum, K. Ahmad, M. A. Ahsan, A. Al-Fuqaha , and J. Qadir, "Collaborative federated learning for healthcare: Multi-modal COVID-19 diagnosis at the edge," 2021.

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

[141] S. Reddi *et al.*, "Adaptive federated optimization," in *Proc. 9th Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=LkFG3lB13U5

[142] A. Reisizadeh, F. Farnia, R. Pedarsani, and A. Jadbabaie, "Robust federated learning: The case of affine distribution shifts," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f5e536083a438cec5b64a4954abc17f1-Abstract.html

[143] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 707–721.

[144] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*.
[145] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," 2018, *arXiv:1811.04017*.
[146] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 57–64.
[147] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–7, doi: 10.1109/GLOCOM.2018.8647927.
[148] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," 2017, *arXiv:1708.08296*.
[149] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter, "Privacy preserving regression modelling via distributed computation," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 677–682.
[150] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *IEEE Netw. Lett.*, vol. 2, no. 1, pp. 23–27, Mar. 2020.
[151] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," 2019, *arXiv:1903.02891*.
[152] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, 1979.
[153] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Comput. Surv.*, vol. 22, pp. 183–236, 1990.
[154] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[155] D. Skvorc, M. Horvat, and S. Srbljic, "Performance evaluation of websocket protocol for implementation of full-duplex web streams," in *Proc. 37th Int. Conv. Inf. Commun. Technol. Electron. Microelectronics*, 2014, pp. 1003–1008.
[156] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
[157] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.
[158] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 245–248.
[159] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2018, pp. 21–28.
[160] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput., Pract. Experience*, vol. 28, pp. 2991–3005, 2016.
[161] L. Su and J. Xu, "Securing distributed machine learning in high dimensions," 2018, *arXiv:1804.10140*.
[162] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan , "Can you really backdoor federated learning?," 2019, *arXiv:1911.07963*.
[163] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. 13th Annu. Conf. Int. Speech Commun. Assoc.*, 2012, pp. 194–197.
[164] M. Swan, *Blockchain: Blueprint for a New Economy*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.
[165] B. Tan, B. Liu, V. Zheng, and Q. Yang, "A federated recommender system for online services," in *Proc. ACM Conf. Recommender Syst.*, 2020, pp. 579–581.
[166] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, pp. 6105–6114, 2019. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html
[167] O. Thakkar, G. Andrew, and H. B. McMahan, "Differentially private learning with adaptive clipping," 2019, *arXiv:1905.03871*.
[168] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.
[169] M. Vartak *et al.*, "MODELDB: A system for machine learning model management," in *Proc. Workshop Hum.-In-the-Loop Data Analytics*, 2016, pp. 1–3.
[170] P. Voigt and A. Von dem Bussche , "The EU general data protection regulation (GDPR)," in *A Practical Guide*, 1st Ed., Cham, Switzerland: Springer International Publishing, 2017.
[171] I. Wagner and D. Eckhoff, "Technical privacy metrics: A systematic survey," *ACM Comput. Surv.*, vol. 51, 2018, Art. no. 57.
[172] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2597–2604.
[173] H. Wang *et al.*, "Attack of the tails: Yes, you really can backdoor federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 16070–16084.
[174] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. 8th Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BkluqlSFDS
[175] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.
[176] R. Wang, H. Li, and E. Liu, "Blockchain-based federated learning in mobile edge networks with application in internet of vehicles," 2021, *arXiv:2103.01116*.
[177] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.
[178] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, "A principled approach to data valuation for federated learning," in *Federated Learn.* Berlin, Germany: Springer, 2020, pp. 153–167.
[179] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.
[180] Y. Wang, "CO-OP: Cooperative machine learning from mobile devices," Univ. Alberta, pp. 1–49, 2019.
[181] Z. Wen, B. He, R. Kotagiri, S. Lu, and J. Shi, "Efficient gradient boosted decision tree training on GPUs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2018, pp. 234–243.
[182] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderSVM: A fast SVM library on GPUs and CPUs," *J. Mach. Learn. Res.*, vol. 19, pp. 797–801, 2018.
[183] Z. Wen, J. Shi, B. He, J. Chen, K. Ramamohanarao, and Q. Li, "Exploiting GPUs for efficient gradient boosting decision tree training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2706–2717, Dec. 2019.
[184] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderGBM: Fast GBDTs and random forests on GPUs," *J. Mach. Learn. Res.*, vol. 21, pp. 1–5, 2020.
[185] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.
[186] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1200–1214, Aug. 2011.
[187] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–19.
[188] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.
[189] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 13–23.
[190] Z. Yan, L. Guoliang, and F. Jianhua, "A survey on entity alignment of knowledge base," *J. Comput. Res. Develop.*, vol. 53, pp. 165–192, 2016.
[191] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, 2019, Art. no. 12.
[192] T. Yang *et al.*, "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*.
[193] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proc. Int. Conf. Wireless Algorithms Syst. Appl.*, 2015, pp. 685–695.
[194] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 603–610.
[195] Z. Yu *et al.*, "Federated learning based proactive content caching in edge computing," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–6.

[196] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7252–7261.

[197] W. Zhang *et al.*, "Blockchain-based federated learning for device failure detection in industrial IoT," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5926–5937, Apr. 2021.

[198] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*.

[199] Z. Zhang, Z. Yao, Y. Yang, Y. Yan, J. E. Gonzalez, and M. W. Mahoney, "Benchmarking semi-supervised federated learning," 2020, *arXiv:2008.11364*.

[200] L. Zhao *et al.*, "Inprivate digging: Enabling tree-based distributed data mining with differential privacy," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 2087–2095.

[201] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing IoT federated learning: A secure, decentralized and privacy-preserving system," 2019, *arXiv:1906.10893*.

[202] Y. Zhao *et al.*, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[203] Y. Zhao *et al.*, "Local differential privacy based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, 2021, doi: 10.1109/JIOT.2020.3037194.

[204] W. Zheng, L. Yan, C. Gou, and F.-Y. Wang, "Federated meta-learning for fraudulent credit card detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, Art. no. 642.

[205] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, pp. 352–375, 2018.

[206] A. C. Zhou, Y. Xiao, Y. Gong, B. He, J. Zhai, and R. Mao, "Privacy regulation aware process mapping in geo-distributed cloud data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1872–1888, Aug. 2019.

[207] P. Zhou, K. Wang, L. Guo, S. Gong, and B. Zheng, "A privacy-preserving distributed contextual federated online learning framework with big data support in social recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 824–838, Mar. 2021.

[208] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

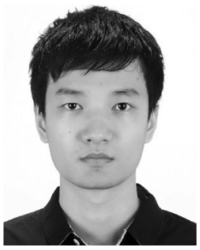
[209] W. Zhuang *et al.*, "Performance optimization of federated person re-identification via benchmark analysis," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 955–963.

[210] G. Zyskind, O. Nathan, and A.'. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, 2015, pp. 180–184.

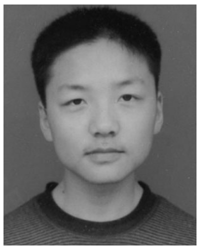
Qinbin Li is currently working toward the PhD degree in the School of Computing, National University of Singapore, Singapore. His current research interests include machine learning, federated learning, and privacy.

Zeyi Wen received the PhD degree in computer science from The University of Melbourne, Australia, in 2015. He is a lecturer of computer science with The University of Western Australia, Australia. His research interests include machine learning systems, automatic machine learning, high-performance computing and data mining.

Zhaomin Wu received the bachelor's degree in computer science from the Huazhong University of Science and Technology, China, in 2019. He is currently working toward the PhD degree in the School of Computing, National University of Singapore, Singapore. His current research interests include federated learning and privacy.



Sixu Hu received the bachelor's degree in computer science from the Huazhong University of Science and Technology, China, in 2019. He is currently working toward the PhD degree in the School of Computing, National University of Singapore, Singapore. His current research interests include federated learning, communication-efficient learning, and benchmarks.



Naibo Wang is currently working toward the PhD degree of data science with the National University of Singapore, Singapore. His current research interests include federated learning, big data analysis and distributed computing.

Yuan Li is currently working toward the PhD degree of data science with the National University of Singapore, Singapore. His current research interests include federated learning, recommender systems.

Xu Liu is currently working toward the PhD degree in the School of Computing, National University of Singapore, Singapore. His current research interests include federated machine learning and recommender systems.

Bingsheng He received the bachelor's degree in computer science from Shanghai Jiao Tong University, China, in 2003, and the PhD degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2008. He is currently an associate professor with the School of Computing, National University of Singapore, Singapore. His research interests are high-performance computing, distributed and parallel systems, and database systems.