Neurocomputing 465 (2021) 371–390

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Federated learning on non-IID data: A survey

Hangyu Zhu [a], Jinjin Xu [b], Shiqing Liu [a], Yaochu Jin [a,*]

[a] Department of Computer Science, University of Surrey, Guildford, Surrey GU2 7XH, UK
[b] Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science of Technology, Shanghai 200237, China

ARTICLE INFO

Article history:
Received 11 June 2021
Revised 9 July 2021
Accepted 15 July 2021
Available online 6 September 2021

Keywords:
Federated learning
Machine learning
Non-IID data
Privacy preservation

ABSTRACT

Federated learning is an emerging distributed machine learning framework for privacy preservation. However, models trained in federated learning usually have worse performance than those trained in the standard centralized learning mode, especially when the training data are not independent and identically distributed (Non-IID) on the local devices. In this survey, we provide a detailed analysis of the influence of Non-IID data on both parametric and non-parametric machine learning models in both horizontal and vertical federated learning. In addition, current research work on handling challenges of Non-IID data in federated learning are reviewed, and both advantages and disadvantages of these approaches are discussed. Finally, we suggest several future research directions before concluding the paper.

1. Introduction

Traditional centralized learning requires all data collected on local devices such as mobile phones to be stored centrally on a data center or cloud server. This requirement not only raises the concern of privacy risks and data leakage, but also poses high demands on storage and computing capacities of the server when the amount of data is huge. Although distributed data parallelism [27], which enables multiple machines to train a model replica with different data groups in parallel, may serve as a potential solution to the issue of storage and computational capacity, it still needs access to the whole training data to split it into evenly distributed shards, causing possible security and privacy problems to the data.

Federated learning (FL) aims to train a global model that can be trained on data distributed on different devices while protecting the data privacy. In 2016, McMahan et al., for the first time, introduce the concept of FL [103] based on data parallelism and proposed a Federated Averaging (FedAvg) algorithm. As a decentralized machine learning approach, FedAvg allows multiple devices to train a machine learning model cooperatively, while keeping the user data stored locally. FedAvg obviates the need for uploading the users' sensitive data to a centralized server, and makes it possible for the edge devices to train a shared model locally within their own local dataset. By aggregating the updates (gradients) of local models, FedAvg meets the basic requirements for privacy protection and data security.

While FL provides a promising approach to privacy protection, many challenges arise in comparison with centralized learning when FL is applied to the real world [161]. These include communication cost needed for transmitting parameters between the server and local devices, computing power and energy consumption required for local devices, and the heterogeneity and randomness of possibly a huge number of local devices in the learning process. A large body of research to address the above challenges, including reducing communication cost [19,103,105,153], FL considering hardware constraints [32,92], and additional protections against adversarial attacks [68,174,185].

Although the authors in [103] claim that FedAvg is able to cope with the not independent and identically distributed (Non-IID) data to a certain degree, a lot of research have indicated that a deterioration in accuracy of FL is almost inevitable on Non-IID or heterogeneous data [181]. The basic assumption of independent and identically distribution in federated learning, which can usually be attributed to the temporal or spatial correlations of data, or the non-stationary distribution of training and test dataset [26], will no longer be satisfied. Non-IID data distributions exist in various machine learning scenarios and tasks. In lifelong learning with different relaxations of the IID assumption, a PAC-Bayesian theorem, is proven to be a generalization of previous IID cases[116]. A novel unsupervised learning method was proposed in [45] for learning and evaluation in the presence of Non-IID label noise. When it comes to FL, the performance degradation can mainly be attributed to weight divergence of the local models

* Corresponding author.
E-mail address: yaochu.jin@surrey.ac.uk (Y. Jin).

https://doi.org/10.1016/j.neucom.2021.07.098
0925-2312/

resulting from Non-IID. That is, local models having the same initial parameters will convergence to different models because of the heterogeneity in local data distributions. During the FL, the divergence between the shared global model acquired by averaging the uploaded local models and the ideal model (the model obtained when the data on the local devices is IID) continues to increase, slowing down the convergence and worsening the learning performance.

Due to the rapid increase of research interest in FL, several valuable review papers on federated learning have been published in the literature. A general introduction to FL and its applications are given in [161,175], detailed discussions of advances and challenges can be found in [68,89]. Analyses of threats and additional privacy preservation techniques in FL are presented in [139,88,9,100]. Overviews of FL applications to IoT and edge devices [80,63,128], wireless networks [60], mobile devices [92], and healthcare [154] have also been reported.

Although Kulkarni et al. [79] have provided a brief introduction to personalization approaches to handling Non-IID data in FL, none of the existing work have explored the impact of Non-IID data on FL in great detail. To fill the gap, this paper is dedicated to a comprehensive survey of FL on Non-IID data, including an in-depth analysis of various data distributions, their influences on model aggregation, a categorization and discussion of pros and cons of existing techniques for handling skewed data distributions, and an outline of remaining challenges and future research in the area of FL in the presence of Non-IID data.

2. Federated learning

FL aims to find an optimal global model θ (Eq. (2)) that can minimize the aggregated local loss function $f_k\left(\theta^k\right)$ (Eq. (1)), where $\boldsymbol{x}$ is the data feature, y is the data label, n_k is the local data size, $n=\sum_{k=1}^{C\times K} n_k$ is the total number of sample pairs, C is the participation ratio assuming that not all local clients participate in each round of model updates, l is the loss function and k is the client index.

$$f_k\left(\theta^k\right)=\frac{1}{n_k}\sum_{i}^{n_k} l\left(\boldsymbol{x}_i, y_i; \theta^k\right) \tag{1}$$

$$\min_{\theta} f(\theta)=\sum_{k=1}^{C\times K}\frac{n_k}{n} f_k\left(\theta^k\right) \tag{2}$$

Generally speaking, FL can be categorized into horizontal and vertical FL according to characteristics of data distribution among the connected clients, which was originally defined in the paper [161]. In the following, we provide a brief introduction to these two FL frameworks before we discussing various data distributions.

2.1. Horizontal federated learning

Horizontal FL is also referred to homogeneous FL [48], which represents the scenarios in which the training data of participating clients share the same feature space but have different sample space. As a simple example shown in Fig. 1, Client 1 and Client 2 contain different rows of data with the same personal features and each row indicates the particular data for one particular person.

The FedAvg algorithm [103] is a typical horizontal FL algorithm and its pseudo code is presented in Algorithm 1, where $m=C\times K$ is the number of participating clients, as stated in [153], a low participation ratio C will lead to the fluctuation of the performance. Both the global model θ_t (where t is the communication round) and all local models θ^k have the same model structure with different model parameter values. Under this assumption, direct model aggregation can be implemented as described in line 9 of Algorithm 1, in which each uploaded local model θ^k under E local epochs using a batch size B and a learning rate η is weighted based on the ratio $\frac{n_k}{n}$, which is proportional to the amount of data on each client, to generate the global model θ_t. Usually, an appropriate setting of E, B and η according to the task may improve the performance of the FL system to a certain degree, which can be determined by sensitivity analysis or optimized by metaheuristic search methods [95,172,173].

Algorithm 1: FedAvg. K is the total numbers of clients; B is the size of mini-batches, T is the total number of communication rounds, E is the total local training epochs, and η is the learning rate.

1: **Server:**
2: Initialize global model θ_0
3: **for** each communication round $t=1,2,\ldots T$ **do**
4: Select $m=C\times K$ clients, where $C\in(0,1)$
5: **for** each **Client** $k=1,2,\ldots m$ in parallel **do**
6: Download θ_t to **Client** k
7: Do **Client** k update and receive θ^k
8: **end for**
9: Update global model $\theta_t \leftarrow \sum_{k=1}^{m}\frac{n_k}{n}\theta^k$
10: **end for**
11:
12: **Client** k update:
13: Replace local model $\theta^k \leftarrow \theta_t$
14: **for** local epoch from 1 to E **do**
15: **for** batch $b\in(1,B)$ **do**
16: $\theta^k \leftarrow \theta^k-\eta\nabla L_k\left(\theta^k, b\right)$
17: **end for**
18: **end for**
19: **Return** θ^k

Compared to the standard centralized learning paradigm, horizontal FL provides a simple yet effective solution to prevent private local data from being leaked, because only the global model parameters θ_t and local model parameters θ^k are allowed to be communicated between the server and clients, and all the training data are kept on the client devices without being accessed by any other parties.

However, frequently downloading and uploading model parameters will consume much communication resources. This becomes even worse if the learning task itself requires large a amount of computation and memory resources like deep learning based monocular depth estimation [106]. Therefore, many techniques have been proposed to reduce communication costs in horizontal FL, such as client updates sub-sampling [131,74,11] and model quantization [153,49,150,105,123]. Besides, Chen et al. [19] suggest to reduce the communication frequency of deep layers of the neural network model to enhance the communication efficiency. In addition, Zhu et al. [183] use a multi-objective evolutionary algorithm to simultaneously increase the model performance and decrease communication costs.

Although no private data can be directly accessed by any third party, the uploaded model parameters or gradients of each client may still leak data information [131], and it has been shown that private image data can be recovered from the gradient information of both shallow and deep neural networks [3,118,145,180,187]. This is because the model gradients intrinsically contain private

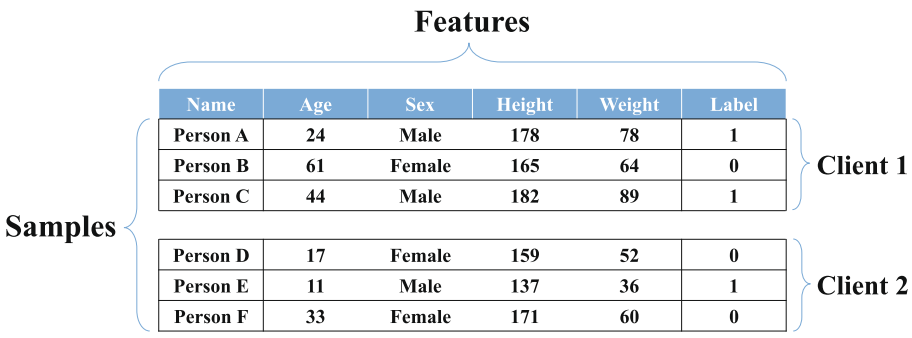


Fig. 1. An illustrative example of data partition in horizontal federated learning. There are two clients, each containing five personal features including name, age, sex, height and weight. Client 1 has the data of Persons A, B, and C, while Client 2 stores the data of Persons D, E and F.

data information and have theoretically been proved to be proportional to the input data [118]. Moreover, an adversarial participating client may secretly construct a local GAN generator [44,57] to generate targeted fake private data samples from other clients. Therefore, additional privacy protection techniques like homomorphic encryption (HE) [5] and differential privacy (DP) [34] have been suggested to address this issue. Phong et al. [118] adopt lattice based additive HE to protect uploaded local model parameters. Hao et al. [51] propose a more efficient symmetric additive HE scheme to reduce the encryption time, while Zhang et al. [174] use introduce an encoding quantization technique to combine the model parameters into a vector and encrypt the generated vector as a whole, further reducing the encryption time in horizontal FL. More recently, Zhu et al. [185] design a distributed additive encryption scheme suited for horizontal FL, in which key pairs are collaboratively generated between the server and clients. On top of the distributed key generation, ternary gradient quantization [150] of the model parameters together with an approximate aggregation method is adopted to significantly save the encryption time, in particular for deep neural networks.

Compared to HE approaches, DP is more favorable in terms of computation time. For instance, local DP is used in [131,41,182,137,125,148], in which the model gradients on the client side are perturbed by adding Gaussian or Laplacian noise before uploading them to the server. Meanwhile, perturbation noise can also be added on the server side to the global model [104,110], known as central DP, to indistinguish the outputs of the aggregation function. Moreover, Truex et al. propose a hybrid privacy-preserving FL scheme by combining Paillier encryption [114] with local DP. The disadvantage of using DP is that the perturbation noise will lead to performance degradation of the global model. Even worse, DP cannot deal with some inverting gradient attacks as introduced in [118,58].

Besides HE and DP based protection methods, other techniques [157,84,163] have also been used, and integrating secure aggregation protocol with a double masking technique [7] is a very popular approach. In this scheme, key pairs are generated by exchanging Diffle-Hellman keys [30] among connected clients and perturbed random numbers can be canceled out after model aggregation on the server. In addition, double masking is robust to the issue caused by offline clients, since the global model parameters can still be retrieved if some clients are offline during training. However, this method applies the Diffle-Hellman key exchange and Shamir secret sharing [127] to every client, which requires a large amount of communication resources.

Finally, horizontal FL often has worse global model performance than centralized learning, especially when parametric models are used in FL. One reason for this is that, horizontal FL needs to do weighted model averaging to update the global model, which has limited theoretical evidence to support the effectiveness of this approach. Although, data parallelism [124,85,12,158,47,126] in distributed machine learning [27,102,71], which has a very similar learning scheme with horizontal FL, has been empirically proved to be valid and widely used in multi-GPU computation to accelerate the learning speed in deep learning, designers need to ensure the training data on multiple devices are evenly distributed. In practice, however, the client training data in horizontal FL are always Non-IID, which may degrade the global model performance.

2.2. Vertical federated learning

Vertical FL [161,96] is also called heterogeneous FL [168], in which users' training data share the same sample space but have different feature spaces. As shown in Fig. 2, Client 1 and Client 2 have the same data samples with different features. Unlike horizontal FL where all clients have their own local data labels, in vertical FL it is often assumed that only one client stores all the data labels (e.g. in Fig. 2, only Client 1 has the data labels and Client 2 has no labels). The client with data labels are called *guest* party [2] (client) or *passive* party [20] (client), whereas the client without data labels are termed *host* party (client) or *active* party (client).

The main steps of vertical FL for training a logistic regression [107] are shown in Algorithm 2, where X_b^k is the batch data on Client k, z_b^k is the inner product of local batch data X_b^k and local model θ_t^k, a is the activation function, $\hat{y}_b$ is the prediction for batch data and $L(y_b, \hat{y}_b)$ is the loss function. For each batch training in line 6 of Algorithm 2, data sample identities of X_b^k must be the same for all clients; otherwise, the prediction $\hat{y}$ cannot be correctly calculated. Different from FedAvg, a horizontal federated learning algorithm, a vertical federated logistic regression system does not contain a central server, nor a shared global model. Both the guest client and host clients have their own local models θ^k corresponding to the feature space of local data X^k, which should be kept locally without sending them to other clients. After calculating the model outputs (inner product) z_b^k on all connected clients, these results will be sent to the guest client for aggregation to calculate the loss function (line 10 Algorithm 2). Finally, the intermediate gradients $\frac{\partial L}{\partial z_b^k}$ can be calculated on the guest client and sent back

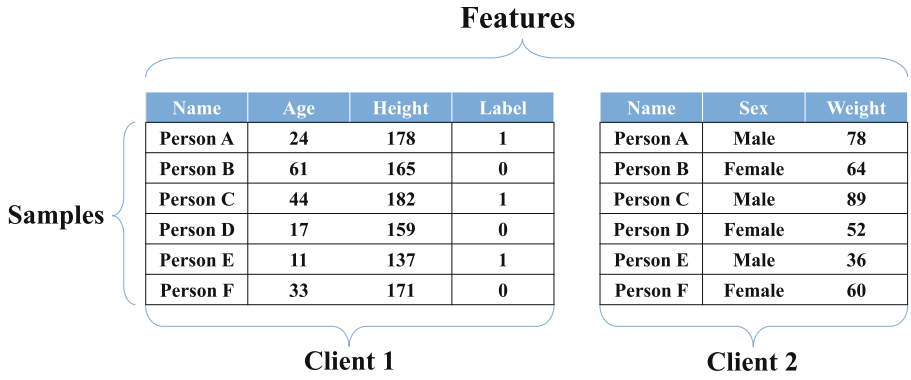


Fig. 2. An illustrative example of vertical federated learning. Client 1 stores three features (name, age, and height) and a label of all six persons, while Client 2 stores two features (sex and weight) only without a label.

to the corresponding host client for local model update. Note that the model on the guest client should also be updated.

Algorithm 2: Vertical FL for logistic regression

1: **Client** 1 is *guest* client and others are *host* clients
2: Training data $X = \left\{X^1, X^2, \dots X^K\right\}$
3: Initialize local model θ_0^k, $k \in (1, K)$
4:
5: **for** each communication round $t = 1, 2, \dots T$ **do**
6: **for** batch data $X_b^k \in \left(X_1^k, X_2^k, \dots X_B^k\right)$ **do**
7: **for** each **Client** $k = 1, 2, \dots K$ in parallel **do**
8: Compute $z_b^k = X_b^k \theta_t^k$ and send it to **Client** 1, if $k \neq 1$
9: **end for**
10: Compute $\hat{y_b} = a\left(\sum_{k=1}^{K} z_b^k\right)$ and $L(y_b, \hat{y}_b)$ on **Client** 1
11: Compute $\frac{\partial L}{\partial z_b}$ on **Client** 1 and send $\frac{\partial L}{\partial z_b^k}$ to **Client** k, $k \in (2, K)$
12: **for** each **Client** $k = 1, \dots K$ in parallel **do**
13: $\theta_t^k \leftarrow \theta_t^k - \eta \frac{\partial L}{\partial z_b^k} \frac{\partial z_b^k}{\partial \theta_t^k}$
14: **end for**
15: **end for**
16: **end for**

Apart from the client training data, there are three main differences between horizontal FL and vertical FL. First, horizontal FL has a server for global model aggregation, while vertical FL does not have this kind of central server, nor a global model. Consequently, the aggregation of local model outputs in vertical FL is performed on the guest client to construct the loss function. Second, model parameters or gradients are communicated between the server and clients in horizontal FL. In vertical FL, by contrast, local model parameters are related to the local data feature spaces that are not allowed to be transferred to other clients. Instead, the guest client receives model outputs from the connected host clients and sends the intermediate gradient values back for local model updates. Finally, the server and clients interact with each other for only once in one communication round in horizontal FL, while the guest client and host clients need to send and receive information for B times in one communication round of vertical FL.

Compared to horizontal FL, training *parametric* models in vertical FL has two advantages. First, models trained in vertical FL always should be in principle have the same performance as the model trained in a centralized way. This is because the computed loss function in vertical FL (line 8 and line 10 of Algorithm 2) is the same as the one in centralized learning. Second, vertical FL often consumes less communication resources than horizontal FL, because only the batch local model outputs z_b^k and intermediate gradients $\frac{\partial L}{\partial z_b^k}$ are required to be transmitted between the guest client and host clients, where z_b^k is a batch set of scalars for logistic regression on client k, and $\frac{\partial L}{\partial z_b^k}$ is the intermediate gradients. Note that the local model θ_b^k can also be used to extract feature representations, and in this case z_b^k becomes a set of vectors whose sizes are determined by the designer. Therefore, the communication costs in vertical FL are determined by the number of data samples and thus vertical FL may consume more communication resources than horizontal FL only if the data size is extremely large.

Privacy preservation is one main challenge in vertical FL. Hardy et al. [53] propose a privacy preserving identity resolution scheme for secure ID alignment in vertical logistic regression, and Nock et al. [112] provide a detailed analysis of the impact of identity resolution. Liu et al. introduce a protocol to protect user sample IDs in asymmetrical vertical FL. Yang et al. introduce a simplified two-party vertical FL framework [162] by removing the third party coordinator. Different from the above work in which parametric models are used for vertical FL, Cheng et al. adopt xgboost [17] decision tree model to construct a secureboost [20] system. Similarly, Wu et al. propose a privacy preserving vertical decision tree learning scheme called Pivot [151], which does not need any trusted third party.

In addition to privacy preservation, effort has also been made to enhance the learning performance in vertical FL. Yang et al. [159] adopt the Quasi-Newton method [117] for vertical federated logistic regression to accelerate the convergence speed. Liu et al. propose a FedBCD algorithm [98], in which each party conducts multiple local updates before communication to reduce the required number of communication rounds. Feng and Yu propose an MMVFL framework [36] to solve multi-class problems with multiple parties. Finally, Chen et al. [18] use a perturbed local embedding technique to simultaneously protect individual's privacy and enhance the communication efficiency in asynchronous vertical FL.

3. Categories of Non-IID Data

The training data on each client in FL heavily depends on the usage of particular local devices, and therefore, the data distribution of connected clients may be totally different with each other. This phenomenon is known as Non-IID [103], which may cause severe model divergence, especially for *parametric* models in horizontal FL. More specifically, for a supervised learning task on client

k, assume each data sample $(\boldsymbol{x}, y)$, where x is the input attribute or feature and y is the label, follows a local distribution $P_k(\boldsymbol{x}, y)$. By Non-IID, we mean P_k differs from client to client. Similar to [68,86], we discuss categories of Non-IID data from the perspective of attribute $\boldsymbol{x}$, label y and other more complicated FL scenarios.

3.1. Attribute skew

Attributes, also known as features or characteristics, usually serve as the input or decision domain $\boldsymbol{x}$ of a machine learning model. Correspondingly, attribute skew indicates the scenarios in which the feature distribution $P_k(\boldsymbol{x})$ across attributes on each client is different from each other. The data attributes across clients can be non-overlapped, overlapped or even the same. Attribute skew varies from vertical FL to horizontal FL, and from problem to problem, too. Generally speaking, attribute skew may include partial overlap and no overlap between the attributes of the data on different clients.

3.1.1. Non-overlapping attribute skew

Non-overlapping attribute skew means that the data features across the clients are mutually exclusive. In this case, if data samples $\boldsymbol{x}$ on different clients k with the same identities hold the same labels $P_k(y|\boldsymbol{x})$, it is known as vertical FL. This non-overlapping property can ensure that the computed total loss of the logistic regression (linear model) in vertical FL is equal to that in centralized learning (refer to line 10 of Algorithm 2). For a dataset like personal information as shown in Fig. 2, client 1 owns the features of age and height, while client 2 has the features of sex and weight. And for image data as shown in Fig. 3, an image of panda is partitioned into two non-overlapped pieces and client 1 stores the left part and client 2 stores the right part. The main difference between these two types of datasets is that the adjacent attributes for personal information may not correlated (for example, 'Age' and 'Height' in client 1 of Fig. 2 have no relationship), but adjacent pixels for image data are always strongly correlated.

3.1.2. Partial overlapping attribute skew

The second category of attribute skew is partial overlapping attribute skew, in which some parts of the data features can be shared with each other. For instance, multi-view images [134,86] are shot from different angles and each party holds single-view (single angle) images. As shown in Fig. 4a, client 1 and client 2 stores the same PC image with different angles. Note that the distribution of each overlapping attribute among different clients may be consistent or inconsistent.

Consistent distribution: In this case, the i-th overlapping attribute x_i is sampled from the same distribution $P(x_i, y)$, which means that the shared attributes will not enlarge the Non-IID divergence of the data.

Inconsistent distribution: As the name suggests, inconsistent distributions of the shared attributes are another reason of Non-IID divergence in addition to the situation in which each client has unique attributes. For example, Xu et al. assume that each client may encounter an infeasible domain when sampling data from the domain of input [155]. One point that needs to be mentioned is, partial overlapping attribute skew may lead to privacy leakage when comparing the shared parts of the attributes from different databases. One typical example is the de-anonymization of Netflix prize challenge dataset [109].

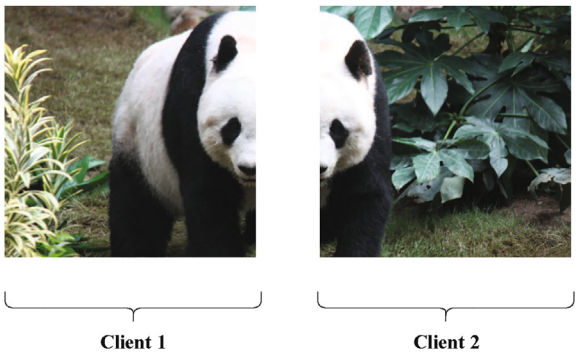


Fig. 3. An example of non-overlapping attribute skew for image datasets, the left half of the image data is stored on Client 1, while the right half of the image is on Client 2.

3.1.3. Full overlapping attribute skew

This case is often known as horizontal FL (horizontal data partition), when the overall data is horizontally *partitioned* among the clients without overlap, as shown in Fig. 1.

Generally, Non-IID divergence of this case usually happens in the presence of inconsistent data distributions, when there is an attribute imbalance of the training data across clients due to perturbations, for example, different degrees of impulse noise across clients, as shown in Fig. 4b. In addition, real world feature imbalance is another feature distribution skew with the same data features. For instance, the EMNIST dataset [21] collects hand written digit numbers from different people, thus, even for the same digit number, the character features (e.g., stroke width and slant) are different.

3.2. Label skew

Label skew represents the scenarios in which the label distribution differ from client to client. There are two slightly different situations of label skew, one is label distribution skew and the other is label preference skew.

3.2.1. Label distribution skew

Label distribution skew is a common Non-IID category, where label distributions $P_k(y)$ on the clients are different and a conditional feature distribution $P_k(x|y)$ is shared across the clients. And it is usually caused by location variations of the clients that store similar types of local training data. Two main kinds of label distribution skew settings [86] have been considered in FL research, namely label size imbalance and label distribution imbalance.

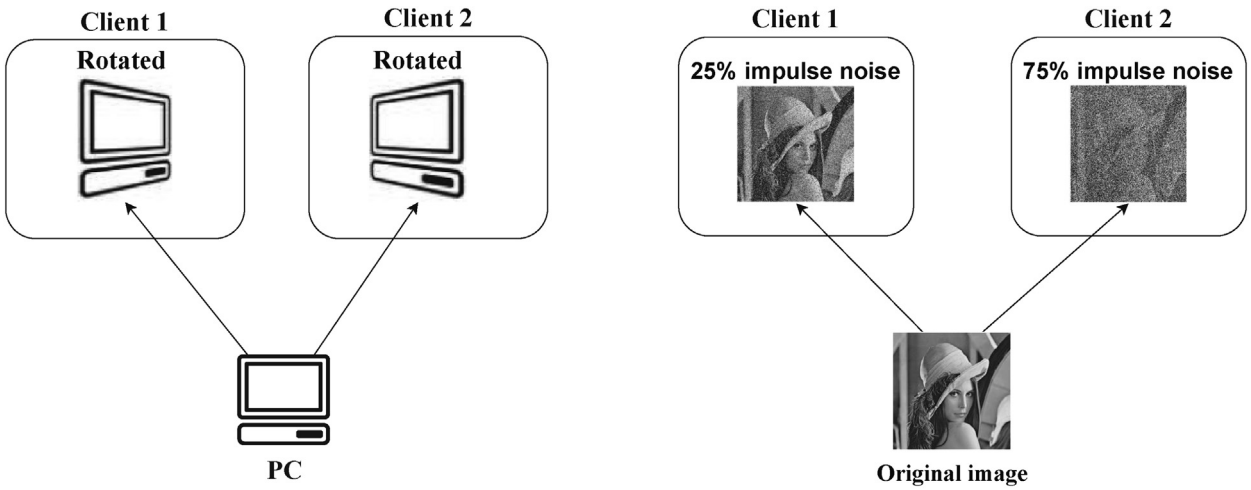
As shown in Fig. 5, label size imbalance is originally proposed in the FedAvg algorithm [103], where each client owns data samples with a fixed number c label classes. c is a hyper parameter that determines the degree of label imbalance, and a smaller c means stronger label imbalance, and vice versa.

Label distribution imbalance is another situation considered in [170], in which a $\boldsymbol{p}_{c,k}$ portion of the instances of label class c is distributed on client k with a probability $p_c \sim Dir_k(\beta)$, where $Dir(.)$ is the Dirichlet distribution and β is the concentration parameter influencing the imbalance level. And a larger β value will result in more unbalanced data partition. Since the imbalance level of data distributions can be easily adjusted by β, some recent studies [87,93,141,142] adopt this method to emulate different Non-IID cases in the real-world.

3.2.2. Label preference skew

Different from label distribution skew, label preference skew considers the client data sample intersection issues often encountered in real-world applications, where the conditional distribution $P_k(y|x)$ may vary across the clients, although $P_k(x)$ is the same.

If the training data across the clients are horizontally overlapped, it is likely that different users annotate different labels



(a) Multi-view of a PC image. (b) Images perturbed by different levels of impulse noise.

Fig. 4. Two examples of attribute distribution skew.

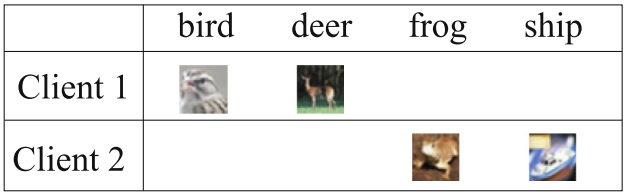


Fig. 5. An illustrative example of label size imbalance with two clients and $c = 2$ using CIFAR10 as an example.

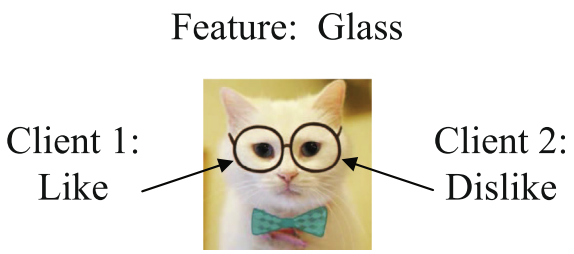


Fig. 6. An example of label preference skew. For the same input feature (a cat with glass), Client 1 may label 'like' while Client 2 may label 'dislike' instead.

for the same data sample due to individuals' preferences. For the image of a cat with glass in Fig. 6, user 1 labels 'like', while user 2 labels 'dislike'.

Crowdsourcing data [38] is a more complex situation where data labels can be noisy, posing serious challenges to information collection in many centralized machine learning tasks, let alone in FL tasks. For example, most local devices only contain unlabelled data and require multiple workers or volunteers to label them. Therefore, it is not uncommon that some labels are incorrect, noisy and even missing.

3.3. Temporal skew

Temporal skew means the skew in distribution in temporal data, including spatio-temporal data and time-series data, also referred to as time-stamped data. This type of data accounts for a large proportion of real-world FL applications (e.g., various measurement data in IoT devices) and hence, it needs to be categorized separately. Different from attribute skew and label skew, temporal skew refers to the inner correlation of data observations in the time domain. Particularly for time series data, the data distributions $P_k(\boldsymbol{x}, y|t)$ (t is the time index) on each client keeps changing over time.

We present an illustrative example of temporal skew, as shown in Fig. 7, where two webcams at different locations take photos of runners over over the time and the deviation of photos on different clients comes from time difference. Although there is a temporal skew in the data collected on Client 1 and Client 2, these data have a great amount of intersections across the entire time period.

In addition, it can happen that different clients collect data during different periods of time rather than the whole time period, which also results in temporal skew. As shown in Fig. 8, Client 1 stores the stock prices for the first 60 months, while Client 2 holds those for the last 60 months.

3.4. Other scenarios

There are some other scenarios that do not belong to any Non-IID categories discussed above.

3.4.1. Attribute & Label skew

In the scenario of Attribute & Label skew, different clients hold data with different labels and different features, which integrates the characteristics of both horizontal and vertical FL.

Specifically, different features can also be extended to a generalized definition that data types may vary across clients. As shown in Fig. 9, client 1 only have image dataset and client 2 only have speech dataset. Global model aggregation becomes extremely hard

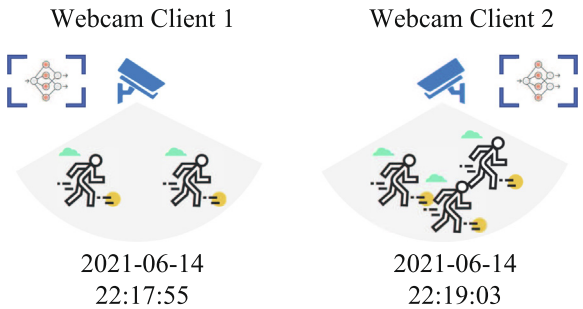


Fig. 7. An example of temporal skew for webcam recordings in FL.

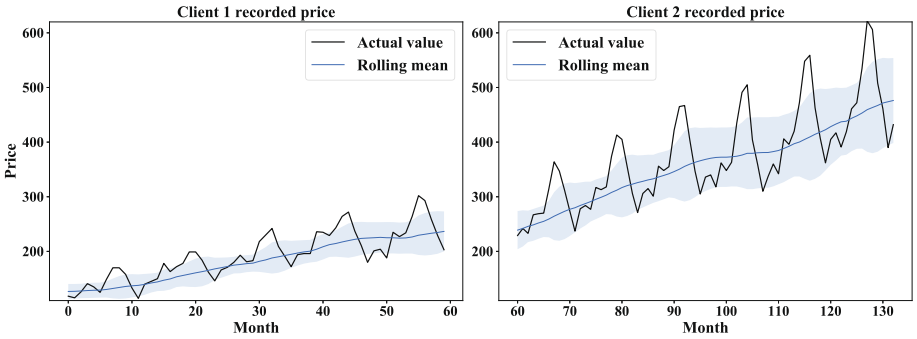


Fig. 8. An example of temporal skew in distributions of stock data in FL.

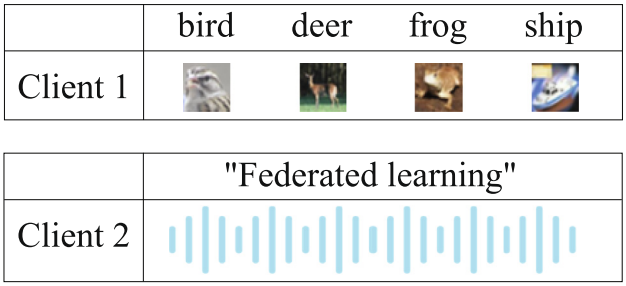


Fig. 9. An example of different features, different labels Non-IID. Client 1 and Client 2 may hold different types of data (image and audio).

in this case, since local model types or structures may be totally different across clients.

3.4.2. Quantity skew

Quantity skew means the number of training data varies across different clients and it can occur in all situations discussed above.

4. Challenges of non-IID data to model training

Both *parametric* models and *non-parametric* models have been used for FL. Due to the differences in the training mechanisms for parametric and non-parametric models in both horizontal and vertical FL, the impacts of Non-IID on their training performances are also considerably different. In this section, we introduce the core training steps of both parametric and non-parametric models in FL at first, and then discuss the influences of Non-IID training data on their performance in horizontal and vertical FL, respectively.

4.1. Horizontal federated learning

If not specified, Non-IID in horizontal FL usually refers to label distribution skew, this is because, label distribution skew often causes more serious client data distribution divergence compared to the case with the same features but different labels. For those extreme cases in which for example, each client contains data samples with only one class [167], it can happen that the global model does not converge at all.

4.1.1. Parametric models

Non-IID data does affect the learning performance on *parametric* models [103,181,123,140,130,8] and will always cause global model divergence in horizontal FL. Since the local data distributions are different from the global data distribution, the averaged local model parameters may also be far away from the global model parameters [181], particularly when the number of epochs for local updates is large [69,121,141]. As shown in Fig. 10, the divergence between the averaged local model parameters θ_{t+1}^{avg} and the actual global model parameters θ_{t+1} are much larger for Non-IID data. In addition, the divergence may accumulate over the communication round t.

Linear models like linear regression and ridge regression are the simplest *parametric* models, which can be directly used in FedAvg algorithm. However, the local private data are more likely to be leaked or reverse-engineered [149,100] for linear models than for other complex parametric models. This is because the communicated gradients of the parameters of linear models are proportional to the local input data [118]. Therefore, additional privacy preserving methods like homomorphic encryption (HE) are usually required to use linear models in horizontal FL.

Neural networks are currently the most widely used *parametric* models in FL, since they have extraordinary performance in computer vision, speech recognition, among others. Three types of neural network models, multi-layer perceptrons [120,39], convolutional neural networks [81] and long-short-term-memory (LSTM) [59], are originally used in the FedAvg algorithm for image classification and next word prediction tasks. Simulation results [103] indicate that FedAvg is able to achieve good performance using *shallow* neural network models on relatively simple datasets, although it performs slightly worse than centralized learning. However, in case *deep* neural networks are used, the FedAvg algorithm becomes very sensitive to client data distribution and may fail to converge on very strongly Non-IID data.

4.1.2. Non-parametric models

Decision trees are typical *non-parametric* models and the gradient boosting decision tree (GBDT) [70] becomes very popular at present in solving both regression and classification problems

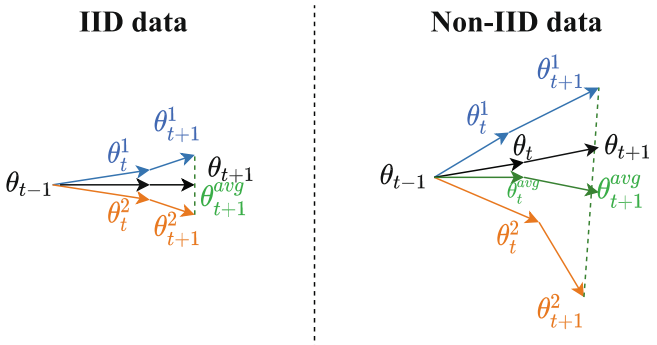


Fig. 10. An illusion of parametric model divergence in horizontal FL for both IID and Non-IID data, where θ_t is the global model and θ_t^{avg} is the averaged model of the local model of Client 1 (θ_t^1) and that of Client 2 (θ_t^2).

because of its good performance. Among gradient boosting tree models, xgboost [17] is the most powerful one, which has already been applied in both horizontal FL [160] and vertical FL [146,20]. The following modifications should be made when the FedAvg algorithm is applied to xgboost:

1. Set binning points (thresholds) for all data features on all connected clients and the server.
2. Each client computes the local histograms for all binning points and send all histograms to the server.
3. The server sums up the received local histograms, and calculates the corresponding impurity values for all binning points. Then, the server will find the best split binning points and send them back to the clients.
4. After getting the best split binning points, clients can split the current node and construct the next-layer of the tree. If stop conditions are not fulfilled, go back to Step 2 and repeat this procedure.

The local histograms are the sum of the local gradients and Hessian matrix for a specific binning point and they intrinsically contain label information of the training data [20], which can be inferred by the server. In order to protect the privacy of the information about the local models, one can often adopt a secure aggregation technique that adds or subtracts an extra random number to the local histograms of each client before sending them to the server. Since the random number is generated by a pseudo random generator with the same seed value, it can be cancelled out with each other [7] after summing the perturbed histograms in the server, as shown in Fig. 11.

Although different horizontally 'partitioned' local training data may lead to different local histograms of every possible split, the summed gradients and Hessian (blue and orange bar in the server of Fig. 11) in the server do not change for each split. Therefore, Non-IID data does not cause any training divergence for decision tree models.

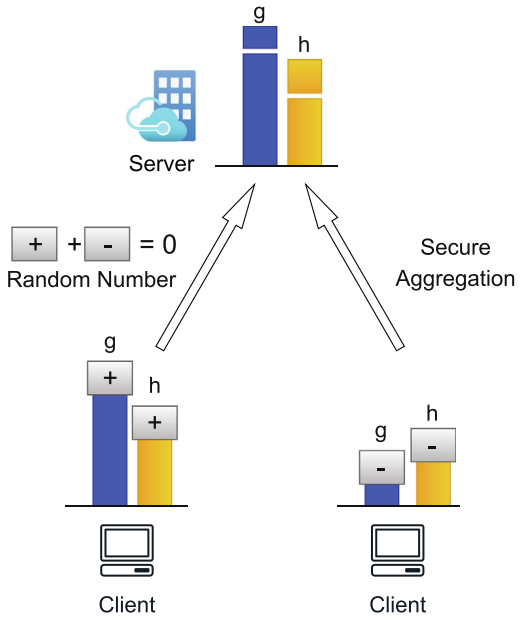


Fig. 11. Secure aggregation used in horizontal federated xgboost, where g is the gradient and h is the Hessian. Secure aggregation [7] is used here for information protection.

4.2. Vertical federated learning

Non-IID in vertical FL usually refers to 'same labels, different features', and most existing work assume that different clients do not have no overlap in features (i.e., no common features) and the data labels are only stored on one client, called the guest client. The rest clients that do not have any labels are known as host clients.

4.2.1. Parametric models

Linear models like logistic regression (generalized linear model) are commonly used in vertical FL. As discussed in Section 2.2 that both model gradients and the training data are stored on local devices during the training period, which reduces both potential data leakage risk and communication costs. To further enhance the security level, some research work [53,96,162] adopts an encrypted model training scheme by encrypting all the model outputs from the host clients. In this case, the gradients of the loss with respect to model parameters can be calculated on ciphertext directly [161] due to the property of HE.

As discussed in Section 2.2, the loss function of training logistic regression in vertical FL has no difference to that in centralized learning. Therefore, Non-IID data does not affect the learning performance for linear models.

Neural networks can also be trained in vertical FL. As shown in Fig. 12, both the guest client K and a host client J own a local neural network model Net_K and Net_J, respectively, which are used to extract features of the local training data. And then the feature representation Z_J on the host client will be sent to the guest client to concatenate with Z_k over the feature dimension. Finally the final output Y_{out} can be generated by passing forward the merged feature output through an neural network model Net_C. There are two main differences between the vertical federated logistic regression and neural network model. First, the local model output z of a single data sample for logistic regression is a scalar, which should be summed up for loss computation. By contrast, the local output Z in the neural network is a vector of feature representation. Second, the guest client needs to construct an extra neural network model to achieve the classification predictions.

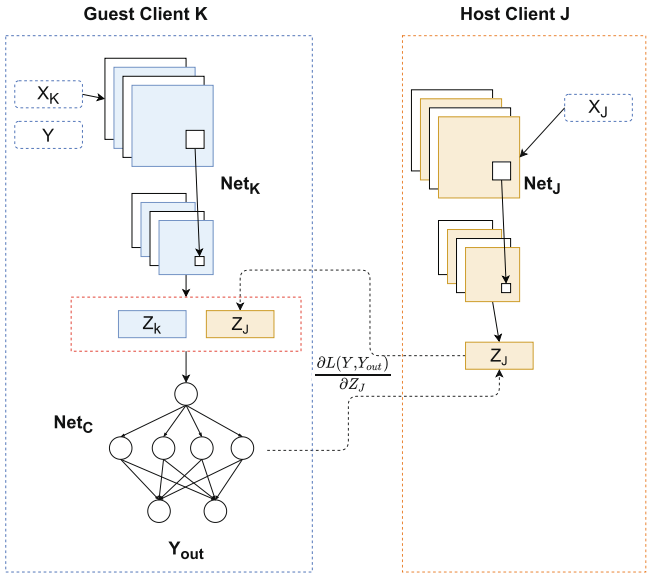


Fig. 12. A simple example for a two-part vertical federated neural network scheme. The local model output Z_J should be sent to the guest client and the merged output Z will be the input of the model Net_C.

It should be noticed that, overall output of the neural networks in vertical FL is different from that in centralized learning, because the neural network is split into several separated sub-networks. A simple example is shown in Fig. 13, where the connections of the global network (denoted by dashed lines) will vanish after splitting it into two sub-networks. In fact, training neural networks in vertical FL remains an open challenge and there are very limited research work focusing on this topic. Recently, Liang et al. present a self-supervised vertical federated neural architecture search to collaboratively fine-tuning both the architecture and model parameters [91]. However, they do not discuss the impact of Non-IID data on the model learning performance.

4.2.2. Non-parametric models

For decision trees in vertical FL, all the histogram values are calculated only on the guest client, because it owns all labels of the training data, as defined in Section 2.2. To avoid leaking data label information of the guest client, all the gradients and Hessian values of node data samples need to be encrypted before sending them to any other host clients. The training steps [20] are as follows:

1. Set binning points of all features for both the guest client and host clients. Initialize key pairs on the guest client.
2. On the current decision node or root node, compute and get the largest impurity value and send the encrypted gradients and Hessian values of node data samples to host clients.
3. For each host client, split data samples for all binning points and sum the corresponding encrypted gradients and Hessian values. Each split is recorded with a unique split number corresponding to the feature and split bin value. The summed values together with the split numbers are sent back to the guest client for decryption.
4. Calculate impurity values for all splits and get the best split with the largest impurity value on the guest client. If the best split is on the guest client, add the feature and split bin value with a unique record ID in the local lookup table. If the best split is on one of host clients, the guest client sends the split number to the corresponding host client. And then this host client can find the feature and split bin value based on the received split number and update the local lookup table.
5. The current decision tree can be constructed with the record ID and client ID. If the stop criterion is not fulfilled, go back to step 2 to construct the next layer of the tree.

A single generated tree model and its corresponding lookup table is shown in Fig. 14. Same as in the aforementioned horizontal scenarios, the learning performance of decision tree models in vertical FL is not influenced by Non-IID data. This is because the calculated impurity values are only dependent on the gradients and Hessian of data samples on the decision node, which has no relationship to data feature distributions among clients.

5. Main approaches to handling non-IID data

As discussed in the last section, Non-IID data, specifically label distribution skew, may cause severe learning divergence to *parametric* models mainly in horizontal FL. However, the FedAvg algorithm *per se cannot deal with model divergence problems caused by Non-IID data, especially when complex models such as neural networks are used in federated learning. Current approaches to dealing with Non-IID problems in horizontal FL can be classified into the data based approach, algorithm based approach, and system based approach, as shown in Fig. 15. Both advantages and disadvantages of these methods will be discussed in detail. In this section, if not specified otherwise, FL means horizontal FL, the baseline FL algorithm is FedAvg described in Algorithm 1, and models are neural networks.*

5.1. Data based approach

Intuitively, the performance degeneration of FL in the presence of Non-IID data is caused by the heterogeneous data distributions and hence, data based approaches aim to address this issue by modifying the distributions. Data sharing and augmentation are two main solutions with state-of-the-art performance.

5.1.1. Data sharing

Data sharing [181] is very straightforward but effective to deal with Non-IID data in horizontal FL. A globally shared dataset G with a uniform distribution is stored on the server and the global model is warmed up by training G. In addition, a random α percentage of G should be downloaded to all connected clients so that the client model is updated by both local training data and the shared global data from G. Experimental results show that the model test accuracy can be enhanced by approximately 30% on the CIFAR10 [78] dataset with only 5% of globally shared data.

Similar ideas are also used in [138,166] to alleviate the negative effect of Non-IID by sharing some local data with the server. Although this data sharing method can significantly enhance the global model performance on Non-IID data (in fact, this approach outperforms most horizontal FL algorithms on Non-IID data), it has very obvious shortcomings. At first, it is hard to get so called uniformly distributed global dataset, since the server has no idea about the data distributions among the connected clients. Second, downloading parts of global dataset to each client for model training violates the requirement of privacy preserving learning, which is the fundamental motivation of FL.

5.1.2. Data augmentation

Data augmentation [136] is originally a technique to increase the diversity of training data by some random transformations or knowledge transfer, which can also be used to mitigate local data imbalance issues in FL. There are three main kinds of data augmentation methods used in FL: the vanilla method, the mixup method

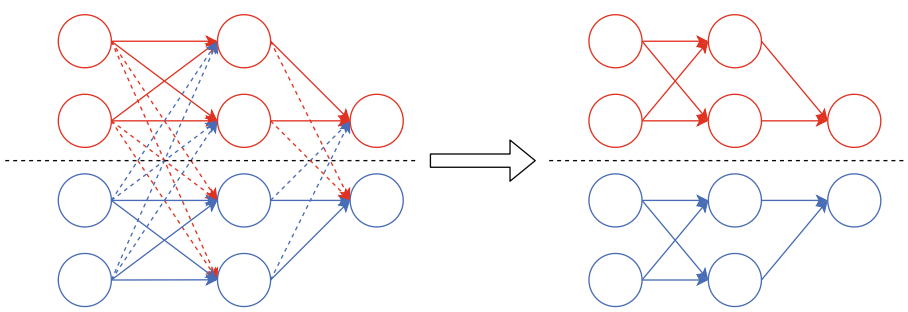


Fig. 13. A simple example for a vertically partition neuron network. The dashed lines denote the coupling connections.

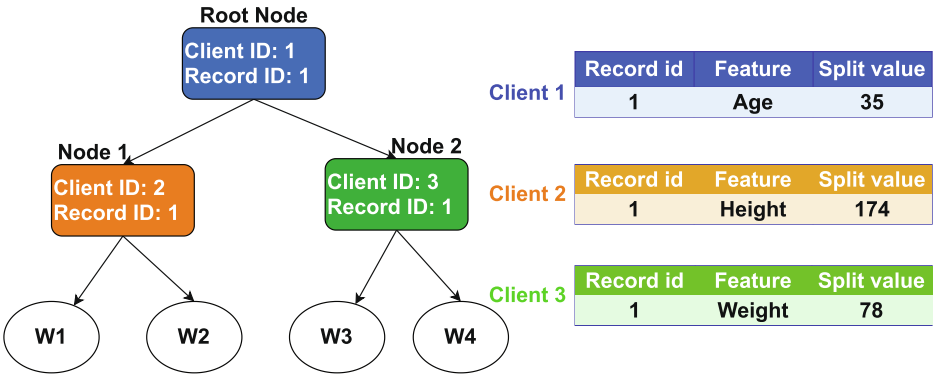


Fig. 14. A simple decision tree built in vertical federated xgboost, where all the decision nodes contain the client ID and record ID only. Both the features and split bin values can be retrieved from the locally stored lookup table. W is the leaf node output.

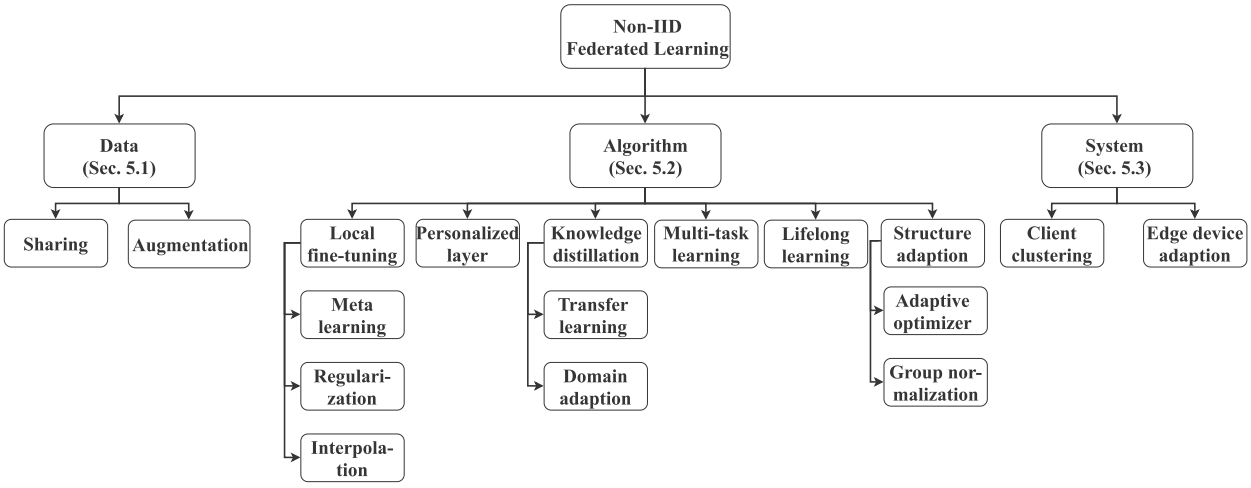


Fig. 15. A summary of existing approaches to addressing Non-IID data.

[176], and the generative adversarial network (GAN) [44] based method.

The basic idea of using the vanilla data augmentation in FL is proposed in [32], in which each client needs to send its label distribution information like the number of data samples for each class c to the server. Then the server can calculate the number of samples C_c for each data class c and their mean value $\overline{C}$. If $C_c < \overline{C}$, client k needs to generate $\left(\overline{C}/C_y\right)^{\alpha}$ number of augmentations for each local data sample (x, y), where α is the hyperparameter to control the degree of augmentation and label y is equal to c. Both the original local data sample and the augmented data are used to update the local model parameters.

The mixup method is another data augmentation approach to tackling Non-IID data. Shin et al. [129] first use this method to propose a XorMixFL framework. The core idea is that each client uploads its encoded seed samples (encoded using the XOR operator) to the server for decoding and the base data samples together with decoded samples in the server can construct a new balanced dataset. After that, a global model is trained on this reconstructed data and downloaded to each client until the training converges. Yoon et al. propose a mean augmented method [165] by exchanging the averaged batch local data with the server. The exchanged mean data will be combined and sent back to each client to reduce the degree of local data imbalance.

Different from the two previous methods, the purpose of federated GAN data augmentation is to train a good generator in the presence of Non-IID data. A general approach is that each client has to send its local seed data samples to the server at first, as does in the data sharing strategy. And then, the server can train the generator and discriminator of GAN based on these seed samples and the well trained generator will be sent to all connected clients. With the received generative model, each client can replenish the training data for the missing labels to construct an IID local dataset. However, sending local data samples to the server violates the data privacy requirement in FL and Yonetani et al. suggest to locally train the discriminators, which will be weighted averaged to construct an aggregated global discriminator on the server, where the weights are determined by the softmax function value calculated across the outputs of discriminators uploaded from connected clients. The generator can be updated by applying the loss gradients computed by the global discriminator. This approach can the protect local data privacy to some extent, but the information of label distribution must be revealed to the server. In order to address this issue, Jeong et al. introduce a multi-hop federated augmentation with sample compression (MultFAug) strategy, which can protect not only data privacy but also the information about label distributions.

Overall, data augmentation techniques can significantly improve the learning performance of the model trained on Non-IID data by replenishing the local imbalanced data with augmentations. However, most of these techniques can only be implemented with the help of aforementioned data sharing, which may increase the risk of data privacy leakage.

5.2. Algorithm based approach

As we stated in Section 2, the goal of federated learning is to collaboratively train a shared model without sharing private data. However, the local models trained by FL may be harmed by the shared model in terms of performance [169], and may fail to generalize due to the drift of heterogeneous data shards [31]. This can be seen from Eq. (1), which indicates that a certain client trains a local model without information exchange between clients, and its model may generalize poorly on unseen data. In addition, the conventional global loss function of a FL system is to minimize Eq. (2), the output of the system is common for all clients and hence, each client may lose their precision on its own task, especially for heterogeneous data or objectives [35]. For example, in a FL system consisting of two clients *A* and *B*, client *A* needs an inference efficiency model and client *B* emphasizes the accuracy. Eventually, it may be difficult to apply the global model to client *A* due to limited computing budget.

To address these issues, personalization approaches have received much attention recently. Personalization, as shown in Fig. 16, aims to adjust the model according to the local tasks. In general, there are several major types of personalization methods, including conducting local fine-tuning (personalization via regularization & interpolation, or meta learning), including a personalization layer, multi-task learning, and knowledge distillation [29].

5.2.1. Local fine-tuning

Local fine-tuning, as the most classic and powerful personalization method, aims to fine-tune the local models after receiving the global model from the server using local data [143], and FedAvg is the basic form of local fine-tuning. The purposes of local fine-tuning are two-folds, i.e., finding a suitable initial shared model, and combining local and global information.

One common idea for fine-tuning is to build a high quality initial global model based on meta-learning methods. A representative method called Personalized FedAvg (Per-FedAvg) [35], which leverages Model-Agnostic Meta-Learning (MAML) [37] to find an initial global model, making it easier for the local clients to obtain good performance with little computation costs. In Per-FedAvg, the authors modify the loss function in Eq. (2) as follows:

$$\min_{w} F(\theta) = \sum_{k=1}^{C\times K} \frac{n_k}{n} f_k(\theta - \alpha \nabla f_k(\theta)), \tag{3}$$

where α is the step size, and the cost function F is the average of meta functions $F_1, F_2, \ldots, F_k, \ldots, F_N$ on client k, which can be denoted by

$$F_k(\theta) = f_k(\theta - \alpha \nabla f_k(\theta)), \tag{4}$$

where the local model θ^k is synchronized with θ and the gradient of local functions can be calculated by

$$\nabla F_k(\theta) = \left(I - \alpha^2 \nabla f_k(\theta)\right) \nabla f_k(\theta - \alpha \nabla f_k(\theta)), \tag{5}$$

where the first and second order information can be replaced by the unbiased estimation using a batch of data. As the authors discussed, the new cost function (Eq. (4)) can capture the differences between clients, and a new client can perform well after being slightly trained based on the obtained initial solution with their own data. This method achieves the first-order optimality with convergence guarantees and better performance on heterogeneous data than FedAvg, but the approximate gradient of Per-FedAvg will significantly affect the results. Similarly, Jiang et al. combine the Reptile algorithm [111] with FedAvg for local personalization [66], however, the performance of their method is only verified on EMNIST-62 and MNIST datasets. Chen *et al.* propose federated meta-learning (FedMeta) framework, where a meta-learner is shared among the clients instead of a global model [16]. Similar to multi-task learning, FedMeta also treats each client as a separated task, and the target is to train a well-initialized model that can be rapidly adapted to any new tasks, but the shared meta-learner may also leads to the leakage of users' privacy.

Combination of local and global information is another approach and several studies on regularization and interpolation have been conducted. The aim of regularization is to minimize the disparity between the global and local models, and FedAvg can be viewed as a special case of regularization-based personalization. Instead of solving Eq. (2) for an explicit global model, Hanzely et al. design a new form of the cost function by adding a regularization term to investigate a trade-off between local and global models [50]. Dinh et al. introduce Moreau Envelopes [108] into the proposed pFedMe algorithm to overcome the statistical

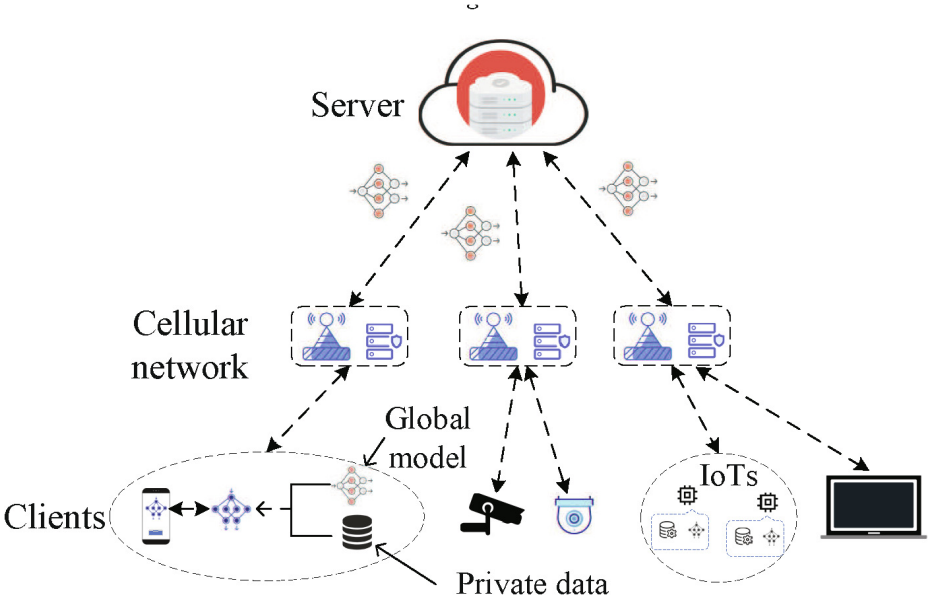


Fig. 16. An illustration of personalization federated learning.

diversity among clients [31]. Specifically, they add a term of l_2 norm for client cost function:

$$f_k(\theta_k) = \frac{1}{n_k}\sum_{i}^{n_k} l(\boldsymbol{x}_i, y_i; \theta_k) + \frac{\gamma}{2}||\theta_k - \theta||^2, \tag{6}$$

where θ is the global model and γ is the regularization parameter. In general, pFedMe outperforms FedAvg on the convergence rate, but there are too many hyperparameters need to be adjusted. Similarly, Huang et al. personalize FL with additional terms and a federated attentive message passing (FedAMP) strategy to relieve the influence of Non-IID data [62], and they guarantee the convergence of the proposed FedAMP and obtains satisfactory results on several widely used datasets.

Interpolation is another idea for combining local and global information, which can further be divided into data interpolation and model interpolation. Data interpolation combines local and global data for training, while model interpolation combines local and global model as the personalized model. Mansour et al. conduct a systematic empirical study on three personalization strategies, client clustering, data interpolation and model interpolation as well as their theoretical guarantees [101]. However, similar to [66], this method need to be examined on more challenging tasks.

5.2.2. Personalization layer

As its name suggests, this type of methods allows each client to have personalized layers in the neural network models. As shown in Fig. 17, each client model consists of personalization layers (filled blocks) and base layers, and only the base layers need to be uploaded to the server for global model aggregation.

A typical paradigm called FedPer is introduced in [4], where the base layers are the shallow layers of the neural network that extracts high-level representations and the personalization layers are the deep layers for classifications. The pseudo code for FedPer is shown in Algorithm 3 and it is clear to see that except for the personalization layers, FedPer is exactly the same as the original FedAvg algorithm. Experimental results have shown that FedPer can achieve much higher test accuracy than FedAvg, especially on strongly Non-IID data. And it is surprising to find that FedPer has achieved better performance on Non-IID data than on IID data.

Algorithm 3: FedPer

Server:
Initialize the shared base model θ_B^0
Initialize personalization layers $\theta_{P_k}^0$
for each communication round $t = 1, 2, \ldots T$ **do**
  Select $m = C \times K$ clients, where $C \in (0, 1)$
  for each **Client** $k = 1, 2, \ldots m$ in parallel **do**
    Download θ_B^t to **Client** k
    Do **Client** k update and receive θ_B^k
  end for
  Update base model $\theta_B^t \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} \theta_B^k$
end for

Client k update:
Merge base model θ_B and personalization layers θ_{P_k}
for local epoch from 1 to E **do**
  for batch $b \in (1, B)$ **do**
    $\left(\theta_B^k, \theta_{P_k}\right) \leftarrow \left(\theta_B, \theta_{P_k}\right) - \eta \nabla L_k\left(\theta_B, \theta_{P_k}; b\right)$
  end for
end for
Return θ_B^k

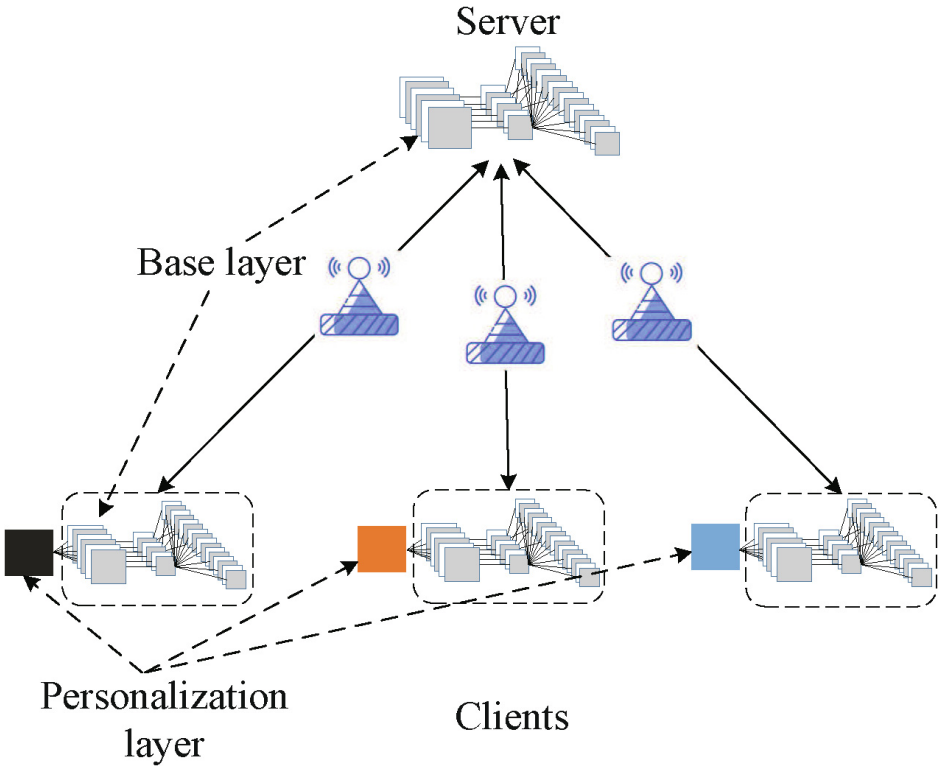


Fig. 17. An illustration of horizontal FL with personalization layers.

By contrast, Liang et al. propose an LG-FEDAVG [90] in which the personalization layers are shallow layers of the neural network and the base layers that are shared with the server are deep layers for class classifications. In addition to supervised local training with the shared global model, LG-FEDAVG also discuss unsupervised learning (autoencoder [77]), self-supervised (jigsaw solving [113]), and adversarial training with an extra locally constructed model connected to the personalization layers. When calculating the test accuracy on new devices, all trained local model logits are averaged before choosing the most likely class. Experiment results show that LG-FEDAVG can achieve much better local test accuracy and slightly better new test accuracy than FedAvg.

Personalization layers are able to not only enhance the learning performance on Non-IID data but also reduce communication costs, since only the base layers rather than the whole model need to be shared between the server and clients. One disadvantage is that each client needs to permanently store the personalization layers without releasing them.

5.2.3. Multi-task learning

An alternative to solving the personalization problem is to treat it as a multi-task learning problem [13]. For example, MOCHA, a representative framework for federated multi-task learning (FMTL), firstly considers issues of communication cost, stragglers and fault tolerance for FL [133]. Due to the use of primal–dual optimization method, MOCHA generates separated but related models for each client, which makes it unsuitable for non-convex optimization tasks. Corinzia et al. propose an FMTL framework VIRTUAL using a Bayesian network and approximated variational inference that can deal with non-convex models. Their method achieves satisfactory results on several Non-IID datasets, but has difficulties in converging when there is a large number of clients, due to the sequential fine-tuning [22]. To alleviate the performance degeneration of FL caused by incongruent data distributions, Sattler *et al.* propose a non-convex FMTL framework, called clustered federated learning (CFL), to group local information [122]. CFL presents a computationally efficient metric of client population distributions based on the cosine similarity, and obtains remarkable results on Non-IID data. However, CFL may pose new challenges to data security due to the reliance on the data similarity.

5.2.4. Knowledge distillation

Knowledge distillation [10,56,46] is also a promising idea for personalized federated learning. The concept of transfer information from large models to small ones is first proposed by Bucilua et al. [10] and popularized by Hinton et al. as knowledge distillation. The main motivation in FL is to transfer knowledge from the server or other clients to a certain client to improve its performance on unknown heterogeneous data. In general, there are two kinds of knowledge distillation strategies adopted in Non-IID FL, namely federated transfer learning and domain adaption.

Transfer learning plays a key role in federated knowledge distillation. In [143], the authors train a next-word prediction model using FedAvg with different hyperparameters for personalization to make sure the model can generalize in new mobile devices. Liu et al. focus on the secure modeling under decentralized data shards and propose a federated transfer learning (FTL) framework, which leverages homomorphic encryption [40] and secret sharing [15] as protocols [97]. Lin et al. [93] utilize an ensemble distillation strategy to robustly fuse multiple models, and the distilled model alleviates the leak risk and computational cost compared to several widely used FL algorithms. The FedDF algorithm is summarized in Algorithm 4.

Algorithm 4: An illustration of the homogeneous FedDF.

1: Initialize server model θ_0, K clients, a total of T communication rounds, n_k samples for a local dataset of client $k \in K$ and its weight p_k, participation ratio C.
2: **for** each communication round $t = 1, 2, \dots T$ **do**
3: $S_t \leftarrow$ random subset (C fraction) of K clients
4: **for** each client $k \in S_t$ **in parallel do do**
5: $\hat{\theta}_t^k \leftarrow$ Local update of FedAvg using θ_{t-1}^k
6: **end for**
7: initialize for model fusion $\theta_{t,0} \leftarrow \sum_{k \in S_t} p_k \hat{\theta}_t^k$
8: **for** j in $\{1, \dots, N\}$ **do**
9: sample a mini-batch of samples $\boldsymbol{d}$, from e.g. (1) an unlabeled dataset, (2) a generator use ensemble of $\left\{\hat{\theta}_t^k\right\}_{k \in S_t}$ to update server student $\theta_{t,j-1}$ through AVGLOGITS
10: **end for**
11: $\theta_t \leftarrow \theta_{t,N}$
12: **end for**
13: **Return** θ_T

From lines 8–10 of the algorithm, we can find that unlabeled data or artificially generated samples are used to assist the knowledge extraction from all participating clients, and the method can be applied to both homogeneous and heterogeneous settings. Compared to the standard FedAvg algorithm, FedDF requires much more computation resources to perform model fusion on the server by using knowledge distillation techniques. However, this model fusion is performed only on the server and has no interference to any clients. On the other hand, FedDF also needs to exchange model parameters between the server and clients, thus, extra protection mechanisms like differential privacy or homomorphic encryption can also be applied. Similarly, Chang et al. propose Cronus, a collaborative robust learning method, by uploading learned features instead of local models to implement local personalization [14]. Li et al. propose a framework called decentralized federated learning via mutual knowledge transfer (Def-KT), in which local clients exchange messages directly in a peer-to-peer manner without the participation of the cloud server [82]. The authors state that the performance degeneration of FedAvg on heterogeneous data may be caused by transferring model parameters only, and the key point of Def-KT is to leverage the advantages of mutual knowledge transfer (MKT) to mitigate the influence of label-shift [178]. Concretely, during each communication round, a subset of selected clients first train their models locally, and then transmit the updated models to the second subset of clients. Then the clients in the second subset can compute two soft predictions (logits) based on their local models and received well trained models. These two calculated logits are used as the dummy labels to update both the local model and received model on each client in the second subset.

Another important chart of knowledge distillation is domain adaption, which emphasizes on eliminating the differences between data shards between clients. A federated adversarial domain adaptation (FADA) algorithm is presented in [115], which uses adversarial adaptation techniques to solve the domain shift problem in FL systems. Li et al. propose FedMD algorithm to enable clients train their unique models on local data [83]. The key element of FedMD is to transfer knowledge from a public dataset (without privacy leakage risk) shared by the clients. For example, the initial model of a certain client is firstly trained on a subset

of CIFAR100 (public set) and then the personalized training on CIFAR10 (private set) will be performed.

5.2.5. Lifelong learning

Lifelong learning is a fundamental challenge in machine learning, where a model is trained on sequential tasks and each task can be seen only once. The main objective is to maintain the model accuracy without forgetting previously learnt tasks. Hence, it is possible to borrow the idea of lifelong learning to overcome the influence of Non-IID data.

Elastic weight consolidation (EWC) [73] is an effective approach to mitigating catastrophic forgetting in lifelong learning, in which the most important parameters for a specific task *A* are identified. When the model is trained on another task *B*, the learner will be penalized for changing these parameters. By making an analogy between federated learning and lifelong learning, Shoham *et al.* propose a federated curvature (FedCurv) algorithm based on EWC as a solution to Non-IID issues in FL [130]. During each round, participants transmit updated models together with the diagonal of the Fisher information matrix, which represents the most informative parameters for the current task. A penalty term is added to the loss function for each participant to promote the convergence towards a globally shared optimum. Furthermore, they assume that the communication cost could be further reduced by a sparse version of the uploaded parameters, without practical verification though. Liu *et al.* combine lifelong and reinforcement learning to form a lifelong federated reinforcement learning (LFRL) architecture [94]. With LFRL, they enable robots to merge and transfer experience, so that the robot can quickly adapt itself to a new environment.

5.2.6. Structure adaption

As mentioned before, training complex *parametric* models like deep neural networks (DNNs) is very challenging in FL especially on Non-IID data. Here we introduce and discuss two useful techniques to accelerate the convergence speed of DNNs in FL.

One technique is to use adaptive optimizers like Adagrad [33], Adam [72], among many others, to replace the standard SGD optimizer. However, these adaptive optimizers often require to accumulate momentum [135] of the previous gradient information for model update, which may double the uploading communication costs in FL. This is because model training is performed only on local devices and the accumulated gradients (with the same size as the model parameters) also need to be uploaded to the server for aggregation. Reddi et al. propose a federated adaptive framework [119] to fix this issue. The core idea is simple: the accumulated gradients are calculated upon the averaged global gradients on the server and each client performs the standard SGD for local model training. This not only saves the local computation resources but also reduces the upload communication costs, since the central server is assumed to be much more powerful than the edge devices.

Another useful technique is to use group normalization (GN) layers [152] to replace the batch normalization (BN) [65] layers in DNNs [179]. BN layers calculate both the mean and variance of one batch training data along the batch dimension during the training and track the exponential moving mean and variance for model prediction. However, in FL, each client device has its own data and the calculated batch moving statistics should also be averaged on the server, which cannot represent the actual global statistics. As a result, local moving statistics are very sensitive to the client data distribution and the aggregated global statistics may fail to converge on Non-IID data [64]. GN is another option by partitioning the channels of each training data into groups and computing per-group statistics separately. Since the statistics of GN are calculated per data sample, it is invariant to the client data distribution. Experimental results have shown that GN exhibits a faster and more stable convergence profile than BN.

5.3. System based approach

5.3.1. Client clustering

Most FL approaches assume that the whole system contains only one global model, which is hard to learn all client information especially in a heterogeneous data environment. Therefore, client clustering is proposed to construct a multi-center framework by grouping the clients into different clusters. And those clients with the similar local training data are allocated to the same cluster. However, the data distribution of each client is private and sensitive information and should be kept secret from the central server. To this end, two main kinds of secure data similarity evaluation methods are introduced in the literature: one is to evaluate the similarity of the loss value, and the other is the similarity of model weights.

The first similarity evaluation approach is reported in [101,75,43,42] by comparing the loss values of different cluster models. The general idea for this technique is straightforward: the server constructs multiple global models instead of a single model and send all cluster models to connected clients for local empirical loss computation. And then each client updates the received cluster model with the smallest loss value and returns it to the server for cluster model aggregation. A representative approach called iterative federated clustering algorithm (IFCA) [42] is described in Algorithm 5, where a one-hot vector $s_{i,j}$ is used to identify the cluster group of the uploaded local model and option I & II are actually the same thing with different global model updating methods. It should be noticed that IFCA reallocates clients to cluster groups based on the calculated local loss of all cluster models across each communication round, making the download payload k times larger than that in FedAvg. To mitigate this issue, Kopparapu and Lin [76] propose a fork algorithm to group clients only at some particular rounds, which reduces the frequency of doing clustering.

Algorithm 5: Iterative federated clustering algorithm (IFCA)

Input: number of clusters k, any single cluster index $j \in [k]$, the total number of communication round T, number of local epochs E, mini-batch size B, learning rate η

for $t = 0, 1, \ldots T-1$ **do**
  Server: Broadcast cluster model $\theta_j^t, j \in [k]$
  Randomly subsample m participating clients
  for client $i \in m$ in parallel **do**
    Determine cluster group: $\hat{j} = argmin_{j\in[k]} F_i\left(\theta_j^t\right)$
    Generate one-hot vector $s_i = \{s_{i,j}\}_{j=1}^k$ with $s_{i,j} = 1\left\{j = \hat{j}\right\}$
    option I (gradient averaging):
      Compute gradient: $g_i = \hat{\nabla} F_i\left(\theta_j^t\right)$
    option II (model averaging):
      $\tilde{\theta}_i = \textbf{ClientUpdate}\left(\theta_j^t, E, B, \eta\right)$
    Send back s_i and g_i or θ_i to the **server**
  end for
  Server:
  option I (gradient averaging): $\theta_j^{t+1} \leftarrow \theta_j^t - \frac{\eta}{m}\sum_{i\in[m]} s_{i,j} g_i$

Algorithm 5: Iterative federated clustering algorithm (IFCA)

17: **option II** (model averaging): $\theta_j^{t+1} \leftarrow \sum_{i\in[m]} s_{i,j}\theta_i / \sum_{i\in[m]} s_{i,j}$
18: **end for**
19: **Return** $\theta_j^T, j \in [k]$
20:
21: **ClientUpdate**$\left(\theta_j, E, B, \eta\right)$ at the i-th machine
22: $\theta^i \leftarrow \theta_j$
23: **for** local epoch from 1 to E **do**
24: **for** batch $b \in [B]$ **do**
25: $\theta^i \leftarrow \theta^i - \eta\hat{\nabla}F_i\left(\theta^i, b\right)$
26: **end for**
27: **end for**
27: **Return** θ^i

The second approach evaluates the local data similarity and does clustering based on the local model weights. Before clustering the clients, some methods [8,122] train and warm up the global model with the standard FedAvg algorithm. And then the warmed-up global model is downloaded to each device for local updating (or just calculating the model gradients) and returned to the server. The server can derive the similarity scores, for instance, the cosine similarity, according to the received model weights and group the clients into clusters based on the calculated similarity scores. In addition, Chen et al. set the fixed cluster clients before training and train cluster models with corresponding clients separately with the FedAvg algorithm. In addition, Xie et al adopt a stochastic expectation and maximization algorithm [28] to perform client clustering and model training at the same time. Instead of a similarity score, the L1 distance between the local model W_i and the cluster model $\widetilde{W}^k$ is computed, as shown in line 4 of Algorithm 6, where m is the total number of clients and K is the total number of clusters. In the expectation step, each local model on client i measure the distance d_{ik} with each global model in cluster k to update cluster assignment r_i^k. In maximization step, all clients are grouped into clusters based on r_i^k (with the shortest distance) and averaging the cluster models. After that, the derived cluster models are downloaded to the clients within the clusters for local model training. Note that the distance (similarity) calculation method can be replaced by other methods like the cosine similarity.

Algorithm 6: FeSEM-Federated stochastic EM

1: Initialize K, $\{W_i\}_{i=1}^m$,$\left\{\widetilde{W}^k\right\}_{k=1}^K$
2: whilestop condition is not satisfied
3: **E-Step:**
4: Compute distance $d_{ik} = Dist\left(W_i, \widetilde{W}^k\right) \quad \forall i,k$
5: Update
$r_i^k = \begin{cases} 1, & \text{if} k = argmin_j Dist\left(W_i, \widetilde{W}^j\right) 0, \quad \text{otherwise} \end{cases}$
6: **M-step:**
7: Group clients into C_k based on r_i^k
8: Update $\widetilde{W}^k = \frac{1}{\sum_{i=1}^m r_i^k} \sum_{i=1}^m r_i^k W_i$
9: **for** each cluster $k = 1, ..K$ **do**
10: **for** $i \in C_k$ **do**
11: Send $\widetilde{W}^k$ to client i

Algorithm 6: FeSEM-Federated stochastic EM

12: $W_i \leftarrow$ **ClientUpdate**$\left(i, \widetilde{W}^k\right)$
13: **end for**
14: **end for**
15: **end while**

Besides, model inference or testing in this scenarios is different from the standard FedAvg algorithm due to multiple cluster models involved. Before calculating the test metrics, all the cluster models need to be downloaded to every clients and a simple method to integrate these calculated metrics is doing (weighted) averaging for different cluster models. Another simple approach requires all clients to compute the loss function values of all cluster models and select the cluster model with the smallest loss for calculating the local test metrics. Apart from these two simple methods, Sattler et al propose an interesting tree-based structure [42] as shown in Fig. 18. At the root node resides, a shared global model θ^* is trained with the standard FedAvg algorithm over all connected clients. In the next layer (the 1st split layer), the client population is split into two clusters based on the cosine similarities, and the cluster models θ_0^* and θ_1^* are trained again by the FedAvg algorithm on the allocated cluster clients. This splitting continues recursively until the stop condition is satisfied. It can be seen that the cluster models near the root node are general models and those located at the bottom layers are personalized models. And different cluster models in the parameter tree can be selected for different applications. Moreover, all the tree models can also be ensembled by the aforementioned two approaches to metrics calculation.

Doing client clustering is necessary and reasonable to deal with Non-IID problems in FL, since aggregating local models with considerably different training data may cause negative knowledge transfer and degrade the shared model performance. In addition, generating multiple global models instead of a single one is beneficial for scalability and flexibility of FL systems so that designers can choose or ensemble different cluster models for specific tasks. However, this method always need to consume additional computation and communication resources for both model training and test.

5.3.2. System level optimization

The fundamental system design of FL is first illustrated by Google [6] and then become very popular in [164]. Google has released its own FL platform called TensorFlow Federated (TFF) [52] to support machine learning with decentralized data. It is built based on the modern machine learning framework Tensorflow [1], which uses 'tf.data' data pipeline to accelerate the learning speed. However, TTF is a single-machine simulation framework and is meant for academic purposes only. WeBank's AI Department has released and maintained an open-source industrial FL framework, called FATE [147], which can support both horizontal and vertical FL with parametric and non-parametric learning models. FATE adopts an efficient session framework Eggroll to implement a high performance distributed computation environment. Additionally, He et al. develop an open-source FL library FedML, in which many state-of-the-art FL algorithms are implemented [55]. FedML uses the mpi4py [24,25] package as the message passing interface to support distributed computation with multiple machines. In terms of eliminating the influence of Non-IID data distribution, Jing *et al.* quantify the performance of federated transfer learning (FTL) using FATE on homogeneous and heterogeneous tasks. And they conclude that inter-process communication, data encryption, internet networking condition of FTL are the main bottlenecks and can be improved further [67]. The performance indicators of these three frameworks are presented in Table 1.

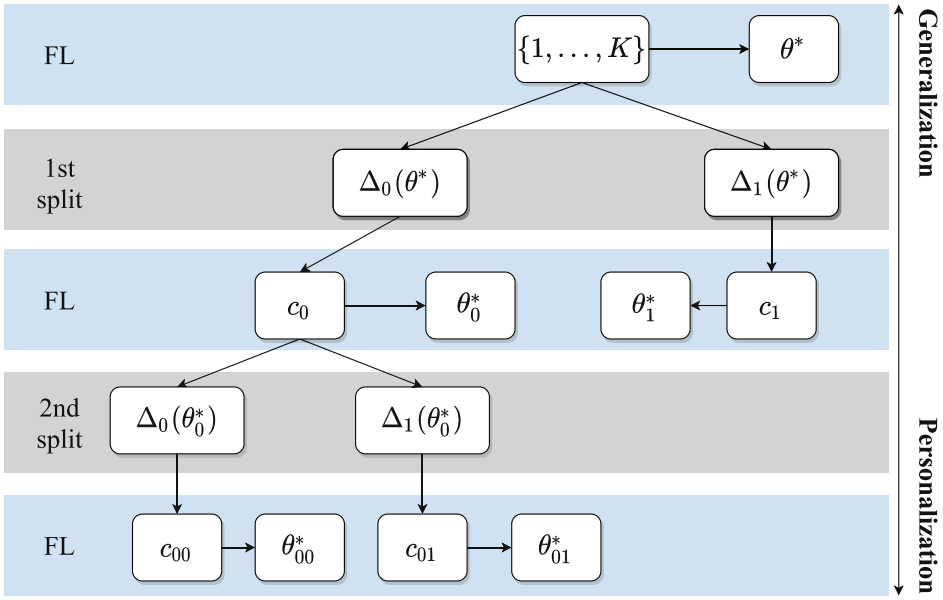


Fig. 18. An exemplary parameter tree created by cluster FL.

Table 1
Performance indicators of Three FL Systemsl.

Framework	Academic use	Industrial use	Communication	Horizontal	Vertical	Encryption	DP
TTF [52]	✓	×	×	✓	×	×	✓
FATE [147]	✓	✓	✓	✓	✓	✓	✓
FedML [55]	✓	✓	✓	✓	✓	×	×

6. Remaining Challenges and Future Directions

Although a large number of remedies have been proposed for handling Non-IID data distributions in FL, many challenges remain open.

- Privacy protection is a basic and vital purpose of FL, but a plenty of methods designed for addressing Non-IID data such as data sharing, knowledge distillation inevitably increase the risk of privacy exposure. It is still not clear to what extent these methods harm the data privacy, and there is no quantitative measures to identify the degree of privacy leakage.
- FL contains a large number of hyperparameters, e.g., the total number of clients, the number of local epochs, and client dropout probability, which strongly differ from algorithm to algorithm, making it hard to benchmark the real Non-IID performance of these algorithms.
- Although two real image dataset are introduced in [99,61], a universal homogeneous and heterogeneous benchmark dataset still lacks for FL. Generate synthetic Non-IID data by arbitrary partitioning datasets cannot effectively evaluate the performance of a method proposed for handling Non-IID data [86].
- Personalized federated learning is promising for IOT edge devices, although it has not yet received enough attention. The system design, model deployment, communication cost reduction in unstable and limited wireless networks and task adaption with limited computation budget remain an open question.
- Although the vertical FL framework is able to be widely adopted in practical industry scenarios, only a handful work has attempted to cope with the potential problems caused by Non-IID distribution under vertical FL settings. For instance, data features with overlapping is one of the most typical cases of Non-IID scenarios for vertical FL, which deserves more attention. In addition, other Non-IID cases with both attribute and label skew, and 'different features, different labels' and 'crowdsourcing skew' have not been fully investigated.
- There is an increasing demand for automated machine learning (AutoML) and lots of algorithms have been presented on neural architecture search (NAS) in practical scenarios [23,144]. However, only a limited amount of research on federated NAS has been reported [156,132,54,184] and little work has considered the influence of Non-IID distributions.

Given the above remaining challenges, we suggest the following future directions:

- Define quantitative criteria for measuring the degree of privacy leakage so that the maximum amount of shared data can be bounded.
- Construct FL benchmark problems reflecting real-world requirements and challenges and provide standard FL hyperparameter settings so that FL algorithms can be fairly compared.
- More cases of Non-IID scenarios for vertical FL need to be further explored, and the corresponding model performance in non-standard Non-IID settings (e.g. with overlapped data features) can be compared to that in standard Non-IID setting. In addition, corresponding algorithms can also be proposed to different Non-IID cases in vertical FL.
- Federated neural architecture search (FNAS) [186] is an emerging research direction and the effect of NON-IID has not been clearly investigated. Therefore, handling Non-IID problems in FNAS is an interesting future direction, and applying heuristic optimization algorithms [172,173] may be a promising solution for issue, since they have been empirically proved to be powerful in evolutionary deep learning [171,177,186].

7. Conclusion

This paper aims to provide a systematic understanding of Non-IID data in federated learning systems and provide a comprehensive overview of existing techniques for handling Non-IID data. A detailed categorization of Non-IID data distributions are given with illustrative examples, several of which have not been discussed in the literature. Different from existing surveys on FL, we focus on the impact of Non-IID data on both parametric and non-parametric models in horizontal and vertical FL. We point out that Non-IID data distributions mainly affect the learning performance of parametric models in horizontal FL and complex models like DNN mor sensitive to client data distributions. We indicate that some existing work on handling Non-IID data, such as local fine-tuning and data sharing achieve better convergence performance often at the cost of either increased local computation and communication resources or even data privacy. Other methods including personalization and client clustering need to change the structure of the vanilla FL framework, making it no longer possible to generate a global model for all clients. Finally, we discuss remaining challenges in handling Non-IID in FL, and suggest a few research directions to handle the open questions.

CRediT authorship contribution statement

Hangyu Zhu: Formal analysis, Investigation, Project administration, Writing - original draft, Writing - review & editing. **Jinjin Xu:** Formal analysis, Investigation, Writing - original draft. **Shiqing Liu:** Formal analysis, Investigation, Writing - original draft. **Yaochu Jin:** Conceptualization, Formal analysis, Methodology, Supervision, Resources, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

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

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] M. Aledhari, R. Razzak, R.M. Parizi, F. Saeed, Federated learning: A survey on enabling technologies, protocols, and applications, IEEE Access 8 (2020) 140699–140725, https://doi.org/10.1109/ACCESS.2020.3013541.

[3] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning: Revisited and enhanced, International Conference on Applications and Techniques in Information Security, Springer. (2017) 100–110.

[4] Arivazhagan, M.G., Aggarwal, V., Singh, A.K., Choudhary, S., 2019. Federated learning with personalization layers. arXiv preprint arXiv:1912.00818.

[5] Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., Strand, M., 2015. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192. https://eprint.iacr.org/2015/1192.

[6] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecný, J., Mazzocchi, S., McMahan, B., Overveldt, T.V., Petrou, D., Ramage, D., Roselander, J., 2019. Towards federated learning at scale: System design, in: Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019, mlsys.org. pp. 374–388.

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

[8] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-iid data, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–9.

[9] Briggs, C., Fan, Z., Andras, P., 2020b. A review of privacy preserving federated learning for private iot analytics. arXiv preprint arXiv:2004.11794.

[10] C. Bucilua, R. Caruana, A. Niculescu-Mizil, Model compression, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 535–541.

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

[12] V. Campos, F. Sastre, M. Yagües, M. Bellver, X. Giró-i Nieto, J. Torres, Distributed training strategies for a computer vision deep learning algorithm on a distributed gpu cluster, Procedia Computer Science 108 (2017) 315–324.

[13] R. Caruana, Multitask learning. Machine learning 28 (1997) 41–75.

[14] Chang, H., Shejwalkar, V., Shokri, R., Houmansadr, A., 2019. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. arXiv preprint arXiv:1912.11279.

[15] W.T. Chang, R. Tandon, On the capacity of secure distributed matrix multiplication, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.

[16] Chen, F., Luo, M., Dong, Z., Li, Z., He, X., 2018. Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876.

[17] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al., Xgboost: extreme gradient boosting, R package version (4-2) (2015) 1.

[18] Chen, T., Jin, X., Sun, Y., Yin, W., 2020. VAFL: a method of vertical asynchronous federated learning. CoRR abs/2007.06081. arXiv:2007.06081.

[19] Y. Chen, X. Sun, Y. Jin, Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation, IEEE Transactions on Neural Networks and Learning Systems 31 (2019) 4229–4238.

[20] Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Yang, Q., 2019. Secureboost: A lossless federated learning framework. arXiv preprint arXiv:1901.08755.

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

[22] Corinzia, L., Beuret, A., Buhmann, J.M., 2019. Variational federated multi-task learning. arXiv preprint arXiv:1906.06268.

[23] Cui, J., Chen, P., Li, R., Liu, S., Shen, X., Jia, J., 2019. Fast and practical neural architecture search, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE. pp. 6508–6517. DOI: 10.1109/ICCV.2019.00661.

[24] L. Dalcín, R. Paz, M. Storti, J. D'Elía, Mpi for python: Performance improvements and mpi-2 extensions, Journal of Parallel and Distributed Computing 68 (2008) 655–662.

[25] L.D. Dalcin, R.R. Paz, P.A. Kler, A. Cosimo, Parallel distributed computing using python, Adv. Water Resour. 34 (2011) 1124–1139.

[26] Darrell, T., Kloft, M., Pontil, M., Rätsch, G., Rodner, E., 2015. Machine learning with interdependent and non-identically distributed data (dagstuhl seminar 15152), in: Dagstuhl Reports, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[27] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M. Ranzato, A.W. Senior, P.A. Tucker, K. Yang, A.Y. Ng, Large scale distributed deep networks, in: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems, Proceedings of a meeting held December 3–6, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 1232–1240.

[28] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. Roy. Stat. Soc.: Ser. B (Methodol.) 39 (1977) 1–22.

[29] Deng, Y., Kamani, M.M., Mahdavi, M., 2020. Adaptive personalized federated learning. arXiv preprint arXiv:2003.13461.

[30] W. Diffie, M. Hellman, New directions in cryptography, IEEE transactions on Information Theory 22 (1976) 644–654.

[31] Dinh, C.T., Tran, N.H., Nguyen, T.D., 2020. Personalized federated learning with moreau envelopes, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual.

[32] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, L. Liang, Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications, in: 2019 IEEE 37th International Conference on Computer Design (ICCD), IEEE, 2019, pp. 246–254.

[33] Duchi, J.C., Hazan, E., Singer, Y., 2010. Adaptive subgradient methods for online learning and stochastic optimization, in: COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27–29, 2010, Omnipress. pp. 257–269.

[34] C. Dwork, Differential privacy: A survey of results, International conference on theory and applications of models of computation, Springer. (2008) 1–19.

[35] Fallah, A., Mokhtari, A., Ozdaglar, A.E., 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual.

[36] Feng, S., Yu, H., 2020. Multi-participant multi-class vertical federated learning. arXiv preprint arXiv:2001.11154.

[37] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning, ICML 2017 Australia, 6–11 August 2017, PMLR. Sydney, NSW, 2017, pp. 1126–1135.

[38] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, V. Verroios, Challenges in data crowdsourcing, IEEE Trans. Knowl. Data Eng. 28 (2016) 901–911.

[39] M.W. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences, Atmospheric environment 32 (1998) 2627–2636.
[40] C. Gentry et al., A fully homomorphic encryption scheme, volume 20, Stanford university Stanford, 2009.
[41] R.C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, 2017, arXiv preprint arXiv:1712.07557.
[42] Ghosh, A., Chung, J., Yin, D., Ramchandran, K., 2020. An efficient framework for clustered federated learning, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual.
[43] Ghosh, A., Hong, J., Yin, D., Ramchandran, K., 2019. Robust federated learning in a heterogeneous environment. arXiv preprint arXiv:1906.06629.
[44] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014(December), pp. 8–13, Montreal, Quebec, Canada, 2014, pp. 2672–2680.
[45] N. Görnitz, A. Porbadnigk, A. Binder, C. Sannelli, M.L. Braun, K. Müller, M. Kloft, Learning and evaluation in presence of non-i.i.d. label noise, in: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014 April 22–25, 2014, JMLR.org. Reykjavik, Iceland, 2014, pp. 293–302.
[46] J. Gou, B. Yu, S.J. Maybank, D. Tao, Knowledge distillation: A survey, Int. J. Comput. Vision (2021) 1–31.
[47] Gupta, V., Choudhary, D., Tang, P.T.P., Wei, X., Wang, X., Huang, Y., Kejariwal, A., Ramchandran, K., Mahoney, M.W., 2020. Fast distributed training of deep neural networks: Dynamic communication thresholding for model and data parallelism. arXiv preprint arXiv:2010.08899.
[48] Haddadpour, F., Mahdavi, M., 2019. On the convergence of local descent methods in federated learning. arXiv preprint arXiv:1910.14425.
[49] Han, S., Mao, H., Dally, W.J., 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings.
[50] Hanzely, F., Richtárik, P., 2020. Federated learning of a mixture of global and local models. arXiv preprint arXiv:2002.05516.
[51] M. Hao, H. Li, G. Xu, S. Liu, H. Yang, Towards efficient and privacy-preserving federated deep learning 2019–2019, in: ICC (Ed.), IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.
[52] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D., 2018. Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604.
[53] Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., Thorne, B., 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint arXiv:1711.10677.
[54] C. He, M. Annavaram, S. Avestimehr, Fednas: Federated deep learning via neural architecture search, in: CVPR 2020 Workshop on Neural Architecture Search and Beyond for Representation Learning, 2020.
[55] He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., Avestimehr, S., 2020b. Fedml: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518.
[56] Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
[57] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: Information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA, 2017, pp. 603–618, https://doi.org/10.1145/3133956.3134012.
[58] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 603–618.
[59] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.
[60] Hosseinalipour, S., Brinton, C.G., Aggarwal, V., Dai, H., Chiang, M., 2020. From federated learning to fog learning: Towards large-scale distributed machine learning in heterogeneous wireless networks. arXiv preprint arXiv:2006.03594.
[61] T.M.H. Hsu, H. Qi, M. Brown, Federated Visual Classification with Real-World Data Distribution, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020.
[62] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, Y. Zhang, Personalized cross-silo federated learning on non-iid data, in: Association for the Advancement of Artificial Intelligence (AAAI), 2021.
[63] Imteaj, A., Thakker, U., Wang, S., Li, J., Amini, M.H., 2020. Federated learning for resource-constrained iot devices: Panoramas and state-of-the-art. arXiv preprint arXiv:2002.10610.
[64] Ioffe, S., 2017. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 1945–1953.
[65] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning ICML 2015, Lille, France, 6–11 July 2015, JMLR.org, 2015, pp. 448–456.
[66] Jiang, Y., Konečný, J., Rush, K., Kannan, S., 2019. Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488.
[67] Q. Jing, W. Wang, J. Zhang, H. Tian, K. Chen, Quantifying the performance of federated transfer learning, ArXiv (2019), abs/1912.12795.
[68] Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al., 2019. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977.
[69] Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T., 2020. SCAFFOLD: stochastic controlled averaging for federated learning, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event, PMLR. pp. 5132–5143.
[70] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T., 2017. Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 3146–3154.
[71] J. Keuper, F.J. Preundt, Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability, in: 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), IEEE, 2016, pp. 19–26.
[72] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings.
[73] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the national academy of sciences 114 (2017) 3521–3526.
[74] Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D., 2016. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492.
[75] K. Kopparapu, E. Lin, Fedfmc: Sequential efficient federated learning on non-iid data, 2020, arXiv preprint arXiv:2006.10937.
[76] Kopparapu, K., Lin, E., 2020b. Fedfmc: Sequential efficient federated learning on non-iid data. CoRR abs/2006.10937. arXiv:2006.10937.
[77] M.A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE journal 37 (1991) 233–243.
[78] Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images.
[79] Kulkarni, V., Kulkarni, M., Pant, A., 2020. Survey of personalization techniques for federated learning, in: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), IEEE. pp. 794–797.
[80] Lan, Q., Zhang, Z., Du, Y., Lin, Z., Huang, K., 2019. An introduction to communication efficient edge machine learning. arXiv preprint arXiv:1912.01554.
[81] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks 3361 (1995) 1995.
[82] Li, C., Li, G., Varshney, P.K., 2020a. Decentralized federated learning via mutual knowledge transfer. arXiv preprint arXiv:2012.13063.
[83] Li, D., Wang, J., 2019. Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581.
[84] Li, H., Han, T., 2019. An end-to-end encrypted neural network for gradient updates transmission in federated learning. arXiv preprint arXiv:1908.08340.
[85] H. Li, A. Kadav, E. Kruus, C. Ungureanu, Malt: distributed data-parallelism for existing ml applications, in: Proceedings of the Tenth European Conference on Computer Systems, 2015, pp. 1–16.
[86] Li, Q., Diao, Y., Chen, Q., He, B., 2021. Federated learning on non-iid data silos: An experimental study. arXiv:2102.02079.
[87] Li, Q., He, B., Song, D., 2020b. Model-agnostic round-optimal federated learning via knowledge transfer. arXiv preprint arXiv:2010.01017.
[88] Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., He, B., 2019. A survey on federated learning systems: vision, hype and reality for data privacy and protection. arXiv preprint arXiv:1907.09693.
[89] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (2020) 50–60.
[90] Liang, P.P., Liu, T., Ziyin, L., Allen, N.B., Auerbach, R.P., Brent, D., Salakhutdinov, R., Morency, L.P., 2020. Think locally, act globally: Federated learning with local and global representations. arXiv preprint arXiv:2001.01523.
[91] Liang, X., Liu, Y., Luo, J., He, Y., Chen, T., Yang, Q., 2021. Self-supervised cross-silo federated neural architecture search. arXiv:2101.11896.
[92] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Communications Surveys & Tutorials 22 (2020) 2031–2063.
[93] Lin, T., Kong, L., Stich, S.U., Jaggi, M., 2020. Ensemble distillation for robust model fusion in federated learning. 34th Conference on Neural Information Processing Systems (NeurIPS 2020).
[94] B. Liu, L. Wang, M. Liu, Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems, IEEE Robotics and Automation Letters 4 (2019) 4555–4562.
[95] J. Liu, Y. Jin, Multi-objective search of robust neural architectures against multiple types of adversarial attacks, Neurocomputing 453 (2021) 73–84.
[96] Y. Liu, Y. Kang, C. Xing, T. Chen, Q. Yang, A secure federated transfer learning framework, IEEE Intell. Syst. 35 (2020) 70–82, https://doi.org/10.1109/MIS.2020.2988525.

[97] Y. Liu, Y. Kang, C. Xing, T. Chen, Q. Yang, A secure federated transfer learning framework, IEEE Intell. Syst. 35 (2020) 70–82.
[98] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T., 2017. Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 3146–3154.
[99] Luo, J., Wu, X., Luo, Y., Huang, A., Huang, Y., Liu, Y., Yang, Q., 2019. Real-world image datasets for federated learning. arXiv preprint arXiv:1910.11089.
[100] Lyu, L., Yu, H., Yang, Q., 2020. Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133.
[101] Mansour, Y., Mohri, M., Ro, J., Suresh, A.T., 2020. Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619.
[102] R. McDonald, K. Hall, G. Mann, Distributed training strategies for the structured perceptron, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Los Angeles, California, 2010, pp. 456–464.
[103] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A., 2017. Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, PMLR. pp. 1273–1282.
[104] McMahan, H.B., Ramage, D., Talwar, K., Zhang, L., 2018. Learning differentially private recurrent language models, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net.
[105] J. Mills, J. Hu, G. Min, Communication-efficient federated learning for wireless edge intelligence in iot, IEEE Internet of Things Journal 7 (2019) 5986–5994.
[106] Y. Ming, X. Meng, C. Fan, H. Yu, Deep learning for monocular depth estimation: A review, Neurocomputing 438 (2021) 14–33, https://doi.org/10.1016/j.neucom.2020.12.089.
[107] D.C. Montgomery, E.A. Peck, G.G. Vining, Introduction to linear regression analysis, John Wiley & Sons, 2021.
[108] J.J. Moreau, Propriétés des applications <<prox>>, Comptes rendus hebdomadaires des séances de l'Académie des sciences 256 (1963) 1069–1071.
[109] Narayanan, A., Shmatikov, V., 2006. How to break anonymity of the netflix prize dataset. arXiv preprint cs/0610105.
[110] Naseri, M., Hayes, J., De Cristofaro, E., 2020. Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy. arXiv preprint arXiv:2009.03561.
[111] Nichol, A., Achiam, J., Schulman, J., 2018. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999.
[112] Nock, R., Hardy, S., Henecka, W., Ivey-Law, H., Patrini, G., Smith, G., Thorne, B., 2018. Entity resolution and federated learning get a federated resolution. arXiv:1803.04035.
[113] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 69–84.
[114] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Advances in Cryptology — EUROCRYPT '99, Springer, Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 223–238.
[115] Peng, X., Huang, Z., Zhu, Y., Saenko, K., 2020. Federated adversarial domain adaptation, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020, OpenReview.net.
[116] Pentina, A., Lampert, C.H., 2015. Lifelong learning with non-i.i.d. tasks, in: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 1540–1548.
[117] B.G. Pfrommer, M. Côté, S.G. Louie, M.L. Cohen, Relaxation of crystals with the quasi-newton method, J. Comput. Phys. 131 (1997) 233–240.
[118] L.T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (2018) 1333–1345.
[119] Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, H.B., 2020. Adaptive federated optimization. arXiv preprint arXiv:2003.00295.
[120] D.W. Ruck, S.K. Rogers, M. Kabrisky, Feature selection using a multilayer perceptron, Journal of Neural Network Computing 2 (1990) 40–48.
[121] Sahu, A.K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., Smith, V., 2018. On the convergence of federated optimization in heterogeneous networks. CoRR abs/1812.06127. arXiv:1812.06127.
[122] F. Sattler, K.R. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, in: IEEE Transactions on Neural Networks and Learning Systems, 2020.
[123] F. Sattler, S. Wiedemann, K.R. Müller, W. Samek, Robust and communication-efficient federated learning from non-iid data, IEEE transactions on neural networks and learning systems 31 (2019) 3400–3413.
[124] F. Seide, H. Fu, J. Droppo, G. Li, D. Yu, 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns, in: Fifteenth Annual Conference of the International Speech Communication Association, 2014.
[125] M. Seif, R. Tandon, M. Li, Wireless federated learning with local differential privacy, in: 2020 IEEE International Symposium on Information Theory (ISIT), IEEE, 2020, pp. 2604–2609.
[126] Shallue, C.J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., Dahl, G.E., 2018. Measuring the effects of data parallelism on neural network training. arXiv preprint arXiv:1811.03600.
[127] A. Shamir, How to share a secret, Commun. ACM 22 (1979) 612–613.
[128] Y. Shi, K. Yang, T. Jiang, J. Zhang, K.B. Letaief, Communication-efficient edge ai: Algorithms and systems, IEEE Communications Surveys & Tutorials 22 (2020) 2167–2191.
[129] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, S.L. Kim, Xor mixup: Privacy-preserving data augmentation for one-shot federated learning, ArXiv (2020), abs/2006.05148.
[130] Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., Zeitak, I., 2019. Overcoming forgetting in federated learning on non-iid data. NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality.
[131] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.
[132] Singh, I., Zhou, H., Yang, K., Ding, M., Lin, B., Xie, P., 2020. Differentially-private federated neural architecture search. arXiv preprint arXiv:2006.10559.
[133] Smith, V., Chiang, C., Sanjabi, M., Talwalkar, A.S., 2017. Federated multi-task learning, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 4424–4434.
[134] H. Su, S. Maji, E. Kalogerakis, E.G. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: 2015 IEEE International Conference on Computer Vision, ICCV 2015, IEEE Computer Society, 2015, pp. 945–953, https://doi.org/10.1109/ICCV.2015.114.
[135] Sutskever, I., Martens, J., Dahl, G.E., Hinton, G.E., 2013. On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013, JMLR.org. pp. 1139–1147.
[136] M.A. Tanner, W.H. Wong, The calculation of posterior distributions by data augmentation, Journal of the American statistical Association 82 (1987) 528–540.
[137] Truex, S., Liu, L., Chow, K.H., Gursoy, M.E., Wei, W., 2020. Ldp-fed: Federated learning with local differential privacy, in: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, pp. 61–66.
[138] Tuor, T., Wang, S., Ko, B.J., Liu, C., Leung, K.K., 2020. Overcoming noisy and irrelevant data in federated learning. arXiv e-prints, arXiv–2001.
[139] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, A. Dubey, No peek: A survey of private distributed deep learning, 2018, arXiv preprint arXiv:1812.03288.
[140] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications IEEE, 2020, pp. 1698–1707.
[141] Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D.S., Khazaeni, Y., 2020b. Federated learning with matched averaging, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020, OpenReview.net.
[142] Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V., 2020c. Tackling the objective inconsistency problem in heterogeneous federated optimization, in: Advances in Neural Information Processing Systems, Curran Associates, Inc. pp. 7611–7623.
[143] Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., Ramage, D., 2019a. Federated evaluation of on-device personalization. arXiv preprint arXiv:1910.10252.
[144] Wang, Z., Lin, C., Sheng, L., Yan, J., Shao, J., 2020d. PV-NAS: practical neural architecture search for video recognition. CoRR abs/2011.00826. arXiv:2011.00826.
[145] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, H. Qi, Beyond inferring class representatives: User-level privacy leakage from federated learning, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications IEEE, 2019, pp. 2512–2520.
[146] Webank, 2019. Fate: an industrial grade federated learning framework. https://fate.fedai.org/.
[147] WeBankFinTech, 2019. Webank: Fate.
[148] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, S. Jin, T.Q. Quek, H.V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.
[149] Wei, W., Liu, L., Loper, M., Chow, K.H., Gursoy, M.E., Truex, S., Wu, Y., 2020b. A framework for evaluating gradient leakage attacks in federated learning. arXiv preprint arXiv:2004.10397.
[150] Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., Li, H., 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 1509–1519.
[151] Wu, Y., Cai, S., Xiao, X., Chen, G., Ooi, B.C., 2020. Privacy preserving vertical federated learning for tree-based models. arXiv preprint arXiv:2008.06170.
[152] Wu, Y., He, K., 2018. Group normalization, in: Proceedings of the European conference on computer vision (ECCV), pp. 3–19.
[153] J. Xu, W. Du, Y. Jin, W. He, R. Cheng, Ternary compression for communication-efficient federated learning, in: IEEE Transactions on Neural Networks and Learning Systems, 2020.
[154] J. Xu, B.S. Glicksberg, C. Su, P. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, Journal of Healthcare Informatics Research 5 (2021) 1–19.
[155] Xu, J., Jin, Y., Du, W., Gu, S., 2021b. A federated data-driven evolutionary algorithm. arXiv preprint arXiv:2102.08288.

[156] Xu, M., Zhao, Y., Bian, K., Huang, G., Mei, Q., Liu, X., 2020b. Neural architecture search over decentralized data. CoRR abs/2002.06352. arXiv:2002.06352.
[157] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, H. Ludwig, Hybridalpha: An efficient approach for privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, 2019, pp. 13–23.
[158] Yadan, O., Adams, K., Taigman, Y., Ranzato, M., 2013. Multi-gpu training of convnets. arXiv preprint arXiv:1312.5853.
[159] Yang, K., Fan, T., Chen, T., Shi, Y., Yang, Q., 2019a. A quasi-newton method based vertical federated learning framework for logistic regression. arXiv preprint arXiv:1912.00513.
[160] Yang, M., Song, L., Xu, J., Li, C., Tan, G., 2019b. The tradeoff between privacy and accuracy in anomaly detection using federated xgboost. arXiv preprint arXiv:1907.07157.
[161] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2019) 1–19.
[162] Yang, S., Ren, B., Zhou, X., Liu, L., 2019d. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. arXiv preprint arXiv:1911.09824.
[163] Yang, X., Feng, Y., Fang, W., Shao, J., Tang, X., Xia, S.T., Lu, R., 2021. Computation-efficient deep model training for ciphertext-based cross-silo federated learning. arXiv:2002.09843.
[164] F. Yin, Z. Lin, Q. Kong, Y. Xu, D. Li, S. Theodoridis, S.R. Cui, Fedloc: Federated learning framework for data-driven cooperative localization and location data processing, IEEE Open Journal of Signal Processing 1 (2020) 187–215.
[165] T. Yoon, S. Shin, S.J. Hwang, E. Yang, Fedmix: Approximation of mixup under mean augmented federated learning, in: International Conference on Learning Representations, 2021.
[166] Yoshida, N., Nishio, T., Morikura, M., Yamamoto, K., Yonetani, R., 2019. Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks. CoRR abs/1905.07210. arXiv:1905.07210.
[167] F. Yu, A.S. Rawat, A. Menon, S. Kumar, Federated learning with only positive labels, International Conference on Machine Learning, PMLR. (2020) 10946–10956.
[168] Yu, F., Zhang, W., Qin, Z., Xu, Z., Wang, D., Liu, C., Tian, Z., Chen, X., 2020b. Heterogeneous federated learning. arXiv preprint arXiv:2008.06767.
[169] Yu, T., Bagdasaryan, E., Shmatikov, V., 2020c. Salvaging federated learning by local adaptation. arXiv preprint arXiv:2002.04758.
[170] Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K.H., Hoang, T.N., Khazaeni, Y., 2019. Bayesian nonparametric federated learning of neural networks, in: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, PMLR. pp. 7252–7261.
[171] N. Zeng, H. Li, Z. Wang, W. Liu, S. Liu, F.E. Alsaadi, X. Liu, Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip, Neurocomputing 425 (2021) 173–180, https://doi.org/10.1016/j.neucom.2020.04.001.
[172] N. Zeng, D. Song, H. Li, Y. You, Y. Liu, F.E. Alsaadi, A competitive mechanism integrated multi-objective whale optimization algorithm with differential evolution, Neurocomputing 432 (2021) 170–182, https://doi.org/10.1016/j.neucom.2020.12.065.
[173] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, X. Liu, A dynamic neighborhood-based switching particle swarm optimization algorithm, IEEE Transactions on Cybernetics (2020) 1–12, https://doi.org/10.1109/TCYB.2020.3029748.
[174] Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y., 2020a. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning, in: 2020 USENIX Annual Technical Conference (USENIX ATC 20), USENIX Association. pp. 493–506.
[175] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021), https://doi.org/10.1016/j.knosys.2021.106775 106775.
[176] Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D., 2018a. mixup: Beyond empirical risk minimization, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net.
[177] H. Zhang, Y. Jin, R. Cheng, K. Hao, Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance, IEEE Trans. Evol. Comput. 25 (2021) 371–385.
[178] Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H., 2018b. Deep mutual learning, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018, IEEE Computer Society. pp. 4320–4328. DOI: 10.1109/CVPR.2018.00454.
[179] Zhang, Z., Yang, Y., Yao, Z., Yan, Y., Gonzalez, J.E., Mahoney, M.W., 2020b. Improving semi-supervised federated learning by reducing the gradient diversity of models. arXiv preprint arXiv:2008.11364.
[180] Zhao, B., Mopuri, K.R., Bilen, H., 2020a. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610.
[181] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V., 2018. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.
[182] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, K.Y. Lam, Local differential privacy based federated learning for internet of things, IEEE Internet of Things Journal. (2020).
[183] H. Zhu, Y. Jin, Multi-objective evolutionary federated learning, IEEE transactions on neural networks and learning systems 31 (2019) 1310–1322.
[184] Zhu, H., Jin, Y., 2020. Real-time federated evolutionary neural architecture search. arXiv preprint arXiv:2003.02793.
[185] Zhu, H., Wang, R., Jin, Y., Liang, K., Ning, J., 2020. Distributed additive encryption and quantization for privacy preserving federated deep learning. arXiv preprint arXiv:2011.12623.
[186] H. Zhu, H. Zhang, Y. Jin, From federated learning to federated neural architecture search: A survey, Complex & Intelligent Systems 7 (2021) 639–657.
[187] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019(December), in: B.C. Vancouver (Ed.), 8–14, 2019, Canada, 2019, pp. 14747–14756.

Hangyu Zhu received B.Sc. and M.Sc. from Yangzhou University, China in 2015, and the RMIT University, Australia in 2017, respectively. He is currently a PhD student with the Department of Computer Science, University of Surrey, working on evolutionary neural architecture search for federated learning.

Jinjin Xu received the B.E. degree from the School of Information Science and Technology, East China University of Science and Technology, Shanghai, China, in 2017. He is currently working towards a PhD. His current research interests include federated learning, data-driven optimization and their applications.

Shiqing Liu received the B.Sc. degree in automation in 2017, and the M.Sc. degree in pattern recognition and intelligent system in 2020, from Beijing Institute of Technology, Beijing, China. She is currently pursuing the Ph.D. degree with a focus on efficient neural architecture search and federated learning in the University of Surrey, Guildford, U.K.

Yaochu Jin received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001. He is currently a Distinguished Chair Professor in Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He was a Finland Distinguished Professor with the University of Jyv?skyl?, Finland, and Changjiang Distinguished Visiting Professor with the Northeastern University, China. His main research interests include evolutionary computation, multi-objective machine learning, secure machine learning, and self-organizing collective systems.
Dr. Jin was a recipient of the 2014, 2016, and 2019 IEEE Computational Intelligence Magazine Outstanding Paper Award, the 2018 and 2020 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He was named 2019-2020 Highly Cited Researchers by the Web of Science Group. Dr Jin is presently the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and Complex & Intelligent Systems. He was an IEEE Distinguished Lecturer for the period from 2013 to 2015 and 2017 to 2019. He is a Fellow of IEEE.