Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Federated Learning: A Survey on Statistical and System Heterogeneity

YU-MIN CHOU[1], and HENG-JAY WANG[2],and FU-CHIANG CHANG[3],and JERRY CHOU[4],
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: ymchou@lsalab.cs.nthu.edu.tw)
[2]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: hengjay.wang@lsalab.cs.nthu.edu.tw)
[3]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: fuchiang@lsalab.cs.nthu.edu.tw)
[4]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: jchou@lsalab.cs.nthu.edu.tw)

Corresponding author: Jerry Chou (e-mail: jchou@lsalab.cs.nthu.edu.tw).

ABSTRACT With the development of Artificial Intelligence (AI) and growing data privacy concerns, Federated Learning (FL), the paradigm for distributed training on data silos in a privacy-preserving manner, is proposed. Specifically, FL allows data silos or edge devices to collaboratively learn a shared model while keeping all the private data locally and thus enable more secure and accurate model training. However, as the FL is deployed in edge devices and applied in large-scale scenarios(e.g., cross-countries), the differences in data properties and computing power lead to the challenges of heterogeneous learning. These issues have quickly attracted the research community's attention but are still open problems and lack systematic review. In this survey, we explore the domain of heterogeneous FL, especially in 1) statistical heterogeneity, non-identically distributed data on edge devices. 2) system heterogeneity, the significant variability in system characteristics on edge devices. We analyze the root causes and the derived issues of the two heterogeneities and propose a unique taxonomy of heterogeneous FL techniques based on their key ideas and strategies. In addition, we present limitations of existing works under multiple heterogeneities and help researchers design algorithms robust to both statistical and system heterogeneities. Finally, we discuss the open problems of the two heterogeneities and several promising future research directions.

INDEX TERMS Federated learning, statistical heterogeneity, system heterogeneity, survey

I. INTRODUCTION

With the vigorous development of edge devices, such as mobile phones and Internet-of-Things (IoT) devices, an unprecedented amount of data are generated at the edge side. While the numerous data are attractive for the deep learning (DL) community, they exist as data silos due to privacy concerns and intolerable communication costs. Specifically, traditional centralized model training approaches require edge devices to transfer their data to a third-party server for training. However, as people gradually attach importance to privacy preservation, legislation such as the general data protection regulation (GDPR) is introduced to prevent the leak of sensitive data, such as health information and personal preferences. In addition, aggregating such a tremendous amount of data is a severe challenge to the communication cost of the server and edge devices. To break data silos and solve the above challenges, federated learning (FL) was proposed and has drawn great attention to academic and industry communities. FL [1] is a distributed learning paradigm without centralizing data and with privacy by default. FL allows data silos (a.k.a. clients), which can be edge devices or organizations, collaboratively train a shared model under the coordination of the FL server. More specifically, the FL server first initializes a global model and broadcasts it to selected clients. Next, each client performs training on the received model by using its private dataset and uploading model updates to the server. Then the server aggregate all model updates to construct an improved global model for the next round of training. The above steps are iterated several times to build a global model which achieves target accuracy. With this innovative operational concept, FL can offer various benefits, such as data privacy enhancement and better utilization of private data. However, the differences between centralized learning and collaborated learning bring several new challenges to FL, including system design [2], communication [3], resource management [4], and privacy attack [5]. In this survey, we focus on the challenge of statistical and system heterogeneity, which has become a critical problem since the larger scale and more extensive learning environment exacerbate the disparity and diversity

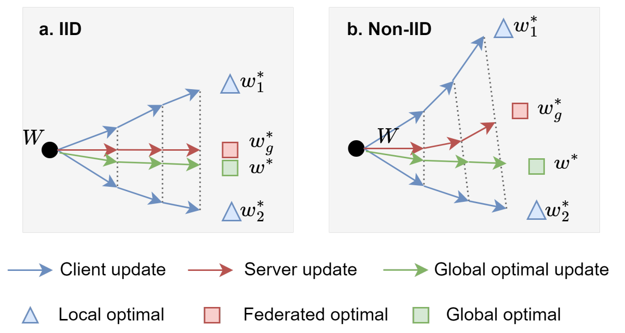

FIGURE 1: Illustration of weight divergence from statistical heterogeneity. The client drift leads to a diverged FL model. (a) IID setting. (b) Non-IID setting

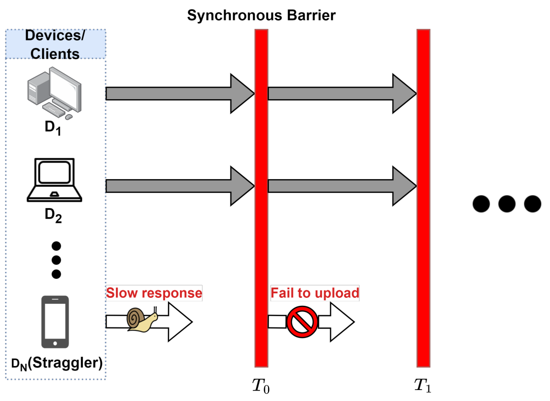

FIGURE 2: Illustration of straggler issue from system heterogeneity. Each round of FL training gets stuck with the straggler.

among the FL clients in terms of their data quality (statistical heterogeneity) and compute capability (system heterogeneity).

A. CHALLENGES OF STATISTICAL HETEROGENEITY

Statistical heterogeneity refers to non-independent and identical distribution (non-IID) data. The IID sampling of training data is an essential assumption for DL algorithms to get an unbiased estimate of the full gradient. Furthermore, the effectiveness of stochastic gradient descent (SGD), widely used in FL, also relies on the IID assumption. In centralized distributed learning, non-IID data have been fine since the server has access to the whole training dataset and splits it in a carefully designed IID manner. Unfortunately, in FL, it is unrealistic to assume that the local datasets are IID since each local dataset is only accessible by the data owner. Moreover, the local datasets usually follow different data distributions due to the difference in how clients collect data. Under the challenge of non-IID data, FL suffers from serious accuracy loss and convergence speed degradation.

B. CHALLENGES OF SYSTEM HETEROGENEITY

System heterogeneity means the significant variability in processing capabilities on edge devices. As the deployments move from the data center to the edge devices, edge devices' computation, and communication capabilities are less powerful than the central server, and the gap between devices is even more prominent. In addition to the heterogeneity of hardware resources, variability in the size of local datasets, network bandwidth, and battery level further contribute to a significantly inefficient training environment. In addition, system heterogeneity makes straggler issue [6] more prevalent than the centralized environment., where straggler issue means a subset of clients may slow down the wall-clock time convergence due to their poor completion time.

Actually, the two types of heterogeneity coexist in FL, and an interplay exists between them. For example, as shown in [7], the methods solving system heterogeneity, such as dropping stragglers (as in FedAvg) or naively tolerating partial model update, implicitly increase statistical heterogeneity and further hurt the model performance. Similarly, the methods solving statistical heterogeneity, such as selecting specific clients in each training round, may hurt training efficiency since the 'good' clients for statistical heterogeneity may be the stragglers, which are 'bad' clients for system heterogeneity. We revisit the concepts in greater detail subsequently.

C. COMPARISON AND OUR CONTRIBUTIONS

With the recent advance of FL, several surveys or tutorials related to FL were proposed. However, the existing studies treat two heterogeneities separately and do not discuss the solutions to the two heterogeneities in depth. For existing surveys in FL, the comparison of their contribution and discussion on heterogeneity are summarized in Table. 1. In this table, we confirm whether the existing survey papers satisfied below four requirements: 1) introduce heterogeneities in detail (background), 2) present and analyze the existing works (algorithm), 3) put two heterogeneities together to discuss (discussion), 4) and proposed a systematic taxonomy (taxonomy).

Some survey papers focus on introducing the concept of FL and summarizing the open challenges. For example, the authors in [8]–[10] provide definitions, categorizations, and applications for FL. Furthermore, they also discuss the open challenges of FL and how FL can be applied to various industries successfully. The authors in [11] discuss the unique characteristics and challenges of FL and introduce a few promising future directions surrounding the challenges of FL. The focus of [12] is systematically introducing FL from the aspects of data partitioning, privacy mechanism, learning model, system architecture, and system heterogeneity. In addition, authors in [2] take a survey on FL from the system view and analyze the system components, including clients, FL server, and the computation-communication framework. Other surveys in [3], [13] discuss the key challenges and future research direction on FL in the context of 5G and 6G wireless communication, respectively. Meanwhile, the authors in [14] discuss FL as an enabling technology for mobile edge computing (MEC) optimization and describe the potential applications of FL in MEC. Furthermore, the survey in [15] presents a concept of integrating FL and blockchain and further explores the opportunities of such integration in MEC networks. In addition, the authors in [4] explore and analyze the opportunity of FL for enabling IoT devices such as data offloading, caching, and privacy. The similar concept of integrating FL and IoT and taxonomy for FL over IoT networks are also explored by [16]. Moreover, the work in [5], [17] both provide comprehensive surveys on threat models and attacks on FL, such as poisoning attacks, inference attacks, and backdoor attacks, and discuss future research directions towards more robust privacy in FL. The authors in [18] explore the concept of personalized FL to address the challenge of statistical heterogeneity.

In summary, the existing surveys and tutorials on FL do not put two heterogeneities together to discuss, and they need more exploration of the interplays between proposed approaches and heterogeneities. Therefore, it motivates us to take a survey that comprehensively discusses system and statistical heterogeneities and how to tackle two heterogeneities simultaneously. The contributions of this survey are as follows.

- We provide an overview, categorization for FL, and a deeper discussion about the challenges of statistical and system heterogeneities.
- To our best knowledge, this survey is the first to perform a holistic analysis of the strategies for solving system and statistical heterogeneities and present limitations of existing works under multiple heterogeneities. Furthermore, we propose a hierarchical taxonomy to present existing works on heterogeneity, as shown in Fig. 3, highlighting the challenges, assumptions, main ideas, and limitations.
- We highlight the lesson learned from the existing works and identify several possible directions for future works toward building FL robust to statistical and system heterogeneity.

The rest of this paper is organized as follows. Section. III reviews the solutions provided to tackling statistical heterogeneity. Section. IV discusses the approaches for system heterogeneity. Section. V discusses the limitations of existing works under multiple heterogeneities and presents the advance works considering both heterogeneities simultaneously. Section. VI highlights several promising research directions for heterogeneity in FL. Section. VII concludes this paper.

II. FEDERATED LEARNING: FUNDAMENTAL CONCEPTS AND PRELIMINARIES

This section presents an introduction to FL, a detailed definition of statistical and system heterogeneities, and an overview of issues brought by the two heterogeneities.

A. FEDERATED LEARNING

FL is a distributed machine learning paradigm aiming to address privacy concerns among data owners. Specifically, FL allows clients to collaboratively train a global model without sharing their private data with others. Therefore, FL has the potential to utilize a large amount of private data and contributes to the progress of deep learning. For an introduction to FL categorizations based on the data distribution characteristics, e.g., vertical, horizontal, and transfer, we refer the readers to [8]. In this survey, if not specified, the FL setting is horizontal.

FL differs from centralized training, which aggregates data from each resource before model training. Generally, the two main kinds of components in the FL system are clients (data owners) and the FL server. Let $\mathcal{N} = \{1, ..., N\}$ denote the set of clients, which can be IoT devices or computers in different organizations. In an FL process, each client $i \in \mathcal{N}$ uses its local private dataset D_i to perform local training on a local model w_i, and uploads only the local model weights to the FL server. Then the FL server aggregates all local models to update a global model w_g.

In general, the FL process follows the below four steps, first proposed by McMahan et al. [1]

1) **Training Initialization**: The FL server first decides on the target task, such as computer vision and natural language processing. Then it sets up the model architectures, data requirements, and hyper-parameters. Some FL frameworks even synchronize hardware resource information with clients in this step.
2) **Client Selection**: Unlike distributed learning, FL allows only a subset of clients S to participate in training in each round instead of all. The most basic approach is random sampling, meaning every client has the same probability of being selected. Moreover, the FL server may select clients based on several factors, such as local loss or communication conditions, to improve model accuracy or training efficiency. After selection, the FL server broadcasts the global model w_g^t to the selected clients, where t denotes the current round index.
3) **Distributed Local Training**: After receiving the global model w_g^t, selected clients start the local training task. Each client $i \in S$ updates the local model w_i^t iteratively by the local dataset D_i and tries to find the optimal model w_i^* that minimizes the loss function $\mathcal{L}(w_i^t)$:
$$w_i^* = \arg\min_{w_i^t} \mathcal{L}(w_i^t)$$
The loss function can be different for different FL approaches. We will discuss this in detail in a subsequent section. Finally, the updated local models $w_{i\in S}$ are subsequently uploaded to the FL server.
4) **Model Aggregation and Global Model Update**: After the server collects local model updates from selected clients, it aggregates them and updates the global model as follows:
$$w_g^{t+1} = \frac{1}{\Sigma_{i\in S}|D_i|} \sum_{i\in S} |D_i| w_i$$
The aggregation method above is called weighted av-

Reference	Topic	Contribution	Heterogeneity			
			Background	Algorithm	Discussion	Taxonomy
[8]	FL concept	Introducing the categorization of different FL schemes, such as horizontal FL, vertical FL, Federated Transfer Learning.	×	×	×	×
[9]		Elaborating comprehensive taxonomies covering various challenge, contributions, and trends.	✓	✓	×	×
[10]		Providing a thorough summary of the most relevant protocols, platforms, and real-life use-cases of FL.	✓	✓	×	×
[11]		Providing a tutorial on FL and a discussion of the open problems worth future research effort.	✓	✓	×	△
[12]		Introducing FL from five aspects: data partitioning, privacy mechanism, machine learning model, communication architecture and systems heterogeneity.	✓	✓	×	×
[2]	FL systems	Providing a comprehensive analysis on FL from a system's point of view.	×	×	×	×
[3]	Wireless FL	Discussing the role of FL addressing the challenges in 5G wireless communication.	×	×	×	×
[13]		Introducing the integration of 6G and FL, FL applications and open problems in the context of 6G communications.	✓	×	×	×
[14]	FL with MEC	Discussing FL as an enabling technology for MEC optimization, and the application of FL in edge computing.	✓	✓	×	×
[15]	FL with blockchain	Discussing an emgerging paradigm in MEC enabled by the integration of FL and blockchain.	✓	✓	×	×
[4]	FL with IoT	Providing a comprehensive survey of FL and IoT networks, especially their integration.	✓	✓	×	×
[16]		Presenting the advances of FL towards enabling FL-powered IoT applications and a taxonomy for FL over IoTnetworks.	✓	✓	×	×
[5]	FL with privacy	Providing a taxonomy covering threat models and major attacks on FL: poisoning attacks and inference attacks.	×	×	×	×
[17]		Providing a study on the security and privacy achievements, issues, and impacts in the FL environment.	×	✓	×	×
[18]	Personalized FL	Exploring personalized FL to address statistical heterogeneity, and presenting a taxonomy of personalization.	✓	✓	×	×

TABLE 1: Summary of selected survey in FL, where ✓ means the objective is well achieved, △ mean that the objective is achieved but can be improved, and × means that the objective is not considered or achieved.

eraging [1], which weights the clients based on the number of samples in their local datasets. In the current research, the aggregation method can also be different for improving model performance.

Then the steps 2∼4 are repeated iteratively until the global model converges or the target number of communication rounds is achieved.

B. STATISTICAL HETEROGENEITY

Statistical heterogeneity refers to the heterogeneity of local datasets across clients, namely, *non-IID*. All along, DL algorithms are usually designed based on the assumption of IID. However, in the FL setting, privacy requirements limit data access to the local datasets. In addition, the data properties of local datasets may be different from each other. After moving away from the IID assumption, FL suffers severe accuracy loss and convergence speed degradation. In this subsection, we first introduce the categories of statistical heterogeneity and then analyze the their impact on FL training. Note that we alternate statistical heterogeneity with non-IID in the following sections.

1) Categories of Statistical Heterogeneity

a: Feature distribution skew

The features refer to the information of data and usually serve as input of a DL model. Feature distribution skew means that the feature distribution $P_i(x)$, where x refers to the input variable, varies between clients i. At the same time, a conditional label distribution $P(y|x)$ is shared, where y refers to the class variable. For example, in a digit handwriting recognition task, different writers can be regarded as different clients. The digits written by different clients may have various strokes, widths, or slants (different feature distribution $P_i(x)$). Meanwhile, the predicted labels of the same digit written by different clients are close (same $P(y|x)$).

b: Label distribution skew

Label distribution skew refers that the label distributions $P_i(y)$ varies between different clients i, while a conditional feature distribution $P(x|y)$ is shared. For instance, in the cat-dog recognition task, the clients preferring cats own more cat data, and vice versa (different label distribution $P_i(y)$). Meanwhile, when the y is given, even the data held by different clients will have the same features (same $P(x|y)$). Generally, the setting of label distribution skew in FL research is making the label distribution of local datasets follow Dirichlet distribution $Dir(\beta)$, where β is the hyperparameter tuning the imbalance level. A larger β value leads

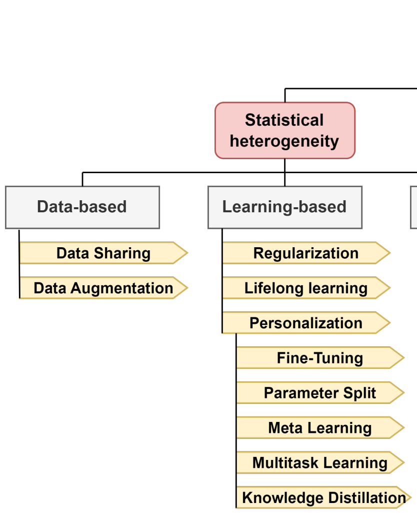
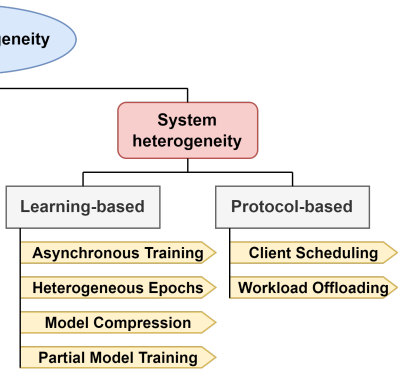

FIGURE 3: Taxonomy of heterogeneous federated learning

to a more unbalanced data partition, and vice versa.

c: Quantity skew

Quantity skew means that the number of local data varies according to clients.

d: Concept shift

Concept shift happens when the relationship between the input and label variable varies between clients. It can be further subdivided into the following two types. The first is that the data sharing the same label may have different features, namely, a conditional feature distribution $P_i(y|x)$ varies between clients, although $P(y)$ is shared. For example, images labeled as 'cat' (same $P(y)$) may contain different species which have distinct features (different $P_i(y|x)$) due to geographic location. The second type of concept shift is that the data sharing the same features may have different labels, namely, a conditional label distribution $P_i(x|y)$ varies between clients, although $P(x)$ is shared. For instance, clients may hold different views (label) on the same image (data) due to their preferences.

2) The Effect of Statistical Heterogeneity on FL

For the above different categories of non-IID, what they have in common is that each local dataset can not represent the global dataset; namely, the data owned by each client is incomplete. Although the authors in [1] claim that FedAvg is robust to non-IID data, recent work [19] has theoretically shown that non-IID data slow down the convergence of FedAvg [1], which match empirical observations. This performance degradation can be attributed to the phenomenon of weight divergence [20], which means that the weights of each local model converge in different directions, and the aggregation of the local models will have an error with the optimal one. Weight divergence can be understood with the illustration in Fig. 1. When data is IID, for clients 1 and 2, the divergence between federated optimal w_g^* (global model), which is the average of local optimal w_i^* (trained local model), and global optimal w^* is small. However, when data are non-IID, the divergence between w_g^* and w^* becomes much larger since the heterogeneity of data makes clients converge to completely different local optimal points. Furthermore, Zhao et al. [20] quantify the weight divergence by the EMD between the distribution over classes on each local dataset. For the mathematical definition of EMD, we refer interested readers to [21]. Here, the important insight is that the greater the difference in the label distribution or feature distribution between local datasets, the higher the degree of non-IID and the higher the impact on training. Although the simulation result shows that sharing data with other clients [20] can reduce EMD and improve accuracy performance, such a data-sharing strategy is unrealistic due to violating privacy requirements.

C. SYSTEM HETEROGENEITY

System heterogeneity means the heterogeneity of processing capabilities on clients. The participants in FL are usually edge devices with less computation and communication capabilities than the powerful clusters. Generally, the time required for each client to complete local training is inversely proportional to its capabilities. Therefore, system heterogeneity leads to variability in the completion time of local training tasks. Then it further brings a straggler issue, which significantly degrades the training efficiency of FL. In this subsection, we first introduce the composition and factors of the local training workload, then detail the straggler issue.

a: Workload of Local Training

Given a local training task, the workload can be divided into the following two types:

- Communication: It includes downloading the latest global model and uploading model updates. The communication workload is positively correlated to the size of the model architecture. For clients, the completion time of communication depends on the stability of network connection, bandwidth, transmission latency, etc.

- Computation: It includes performing training on the local model with private data. The computation workload is positively correlated to the complexity of model architecture, the number of local epochs, size of local datasets. For clients, the completion time of computation depends on the hardware resources, such as processing units, memory, battery level, etc.

b: Straggler Issue

Synchronous distributed learning means the server must wait for every worker to complete assigned work before starting the next round of training. If the difference in workers' completion time is too large, a straggler issue will be derived. Specifically, as shown in Fig. 2, the straggler issue means that the time cost of a training round is bounded by the slowest worker and significantly reduces the training efficiency. Unfortunately, the standard FL algorithm is synchronous, and system heterogeneity causes the completion time of clients to vary a lot. In FedAvg, the way of handling system heterogeneity is dropping stragglers which do not complete local training in time. Although the exclusion of stragglers improves training efficiency, it makes the waste of potential high-quality data and may bring model performance reduction. For FL, handling stragglers or reducing the computation and communication costs without hurting model performance is still an open question.

D. BASIS AND ADVANTAGES OF OUR TAXONOMY

The optimization of FL involves many aspects, such as adjustment of local datasets, local training algorithms, aggregation methods, and interaction between FL components. In order to help readers better understand which part of the FL process each work optimizes, we derive a taxonomy based on *where* the works optimizes. As shown in Fig. 3, we divide existing works into data-based, learning-based, and protocol-based. Data-based algorithms are usually executed during task initialization (step 1 in Sec II-A), and they all focus on augmenting local datasets. Next, the learning-based category refers to the works optimizing learning algorithms of FL, such as the DL hyper-parameters, local training (step 3 in Sec II-A), and aggregation methods (step 4 in Sec II-A). Last but not least is the protocol-based category, which refers to the works optimizing the interaction between the FL server and clients, such as client selection (step 2 in Sec II-A). Moreover, some works in the protocol-based category even try to modify standard FL architecture to achieve more efficient training.

After categorizing existing works based on *where* they optimize, we further classify them based on *how* they optimize, namely, their key techniques. For example, the data-based approaches are divided into data sharing and data augmentation based on whether they directly share raw data. More detailed discussions with insights will be given in later sections. In summary, our taxonomy can help readers not only understand what state-of-the-art methods are but also better understand which parts of the FL process these approaches specifically optimize. It is worth noting that a single work can be classified into more than one category since it involves multiple optimization methods. For example, Astraea [22] is classified into data-based and protocol-based approaches since it involves data augmentation techniques and client scheduling algorithms.

III. TAXONOMY OF STATISTICAL HETEROGENEITY

In this section, we present a unique taxonomy of works on statistical heterogeneity according to their key insight and technique. Based on our proposed taxonomy, the works are divided into *data-based*, *learning-based*, and *protocol-based*. Data-based algorithms aim to solve statistical heterogeneity by modifying the data distribution of local datasets to reduce the degree of non-IID. In addition, learning-based algorithms revise the optimization objective or the learning algorithm to relieve the negative impact brought by non-IID data. Finally, protocol-based algorithms solve it by proposing new FL system architectures or communication protocols. The detailed discussions are as follows.

A. DATA-BASED ALGORITHMS

Motivated by the fact that statistical heterogeneity arises from the EMD between the data distribution of local datasets, data-based algorithms aim to re-balance the data distributions of local datasets and make them more representative of the global data distribution. Specifically, data-based algorithms can be divided into two categories. The first is directly sharing raw data with clients or the server, and the other is performing data augmentation to compensate for the lack of specific data types. We detail the data-based algorithms in this subsection.

1) Data Sharing

As shown in [20], the test accuracy falls sharply with respect to EMD beyond a certain threshold. Thus, for non-IID data, the model performance can significantly increase by slightly reducing EMD. The intuitive and effective way to achieve it is by building an IID globally shared dataset. As shown in Fig. 4(a), Zhao et al. [20] proposed a data-sharing strategy that distributed a subset of the globally shared dataset to each client to re-balance its data distribution. The experiment results show that the accuracy can be improved by 30% for the CIFAR-10 dataset with only a tiny amount of globally shared data. In contrast to distributing data to clients, Hybrid-FL [23] updates the global model with the shared dataset gathered from the volunteered clients in addition to aggregating the model updates in each round. Hybrid-FL can be intuitively seen as several additional clients with IID data also participating in each round of training, and the EMD is therefore reduced. Although the above two algorithms differ in applying globally shared datasets, they achieve the same purpose. Both works show that a shared dataset with a size of 1%∼5% of the total data is enough to relieve the accuracy

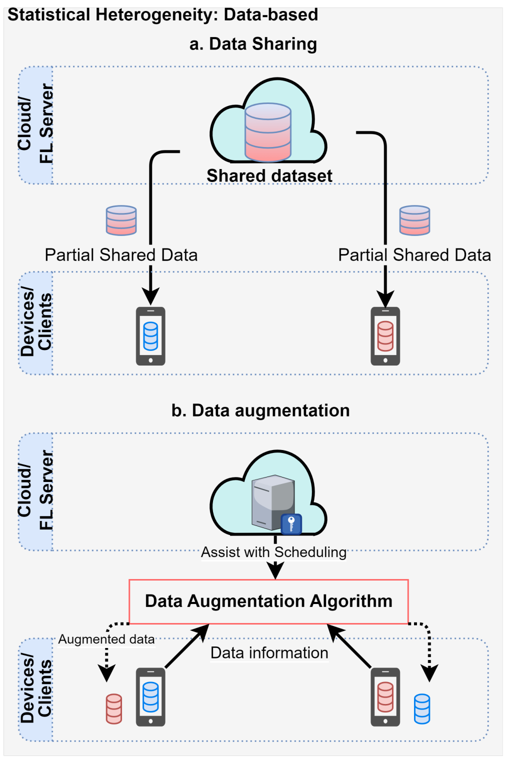

FIGURE 4: Illustration of data-based approaches, including (a) data sharing and (b) data augmentation.

degradation brought by non-IID data effectively. However, it is unrealistic to assume that the server already has an available globally shared dataset [20]. In addition, requiring the clients to upload raw data [23] also violates the purpose of FL.

2) Data Augmentation

Compared with data sharing that directly shares raw data, data augmentation reduces EMD through generating synthetic data (Fig. 4(b)). Data augmentation has been extensively studied in DL to compensate for the lack of data. For class-imbalance learning, over-sampling [24], [25] which generates synthetic data for the minority class, and under-sampling [26] which uses only a subset of the majority class, have been proposed. In addition, data augmentation techniques based on the original data, such as cropping, rotation, and random erasing, are also used to inflate the dataset and further improve the performance of the DL model [27]–[29]. However, the above techniques can not be directly applied to FL since the private data is only accessible to the data owner.

Limited by the privacy requirement of FL, data augmentation is highly challenging in FL. The data augmentation algorithms in FL can only rely on a tiny number of local data or conceive some forms of data sharing that do not violate privacy. Duan et al. [22] proposed Astraea, a self-balancing FL framework that solves imbalanced global data by Z-score-based data augmentation. Astraea first defines the under-sampling and over-sampling threshold and treats a class as a majority or minority class if its z-score is greater than the under-sampling or less than the over-sampling threshold. Then all clients perform data augmentation, such as shift, rotation, and zoom, on minority classes and drop partial samples of majority classes. However, the augmentation methods in Astraea rely on a tiny local dataset of each client, so it may not be feasible in highly skewed non-IID scenarios. In addition, it may take work to decide the thresholds for different scenarios in advance. To generate more diverse data and tackle the limitation of relying on a single local dataset, several works indirectly share data information. Hao et al. [30] proposed ZSDG that generates synthetic labeled data by finding the data that result in similar statistics as those stored in the batch normalization layers of the trained model. Although ZSDG does not need any access to private data and can generate diverse data, the quality of synthetic data highly depends on the model performance. Therefore, ZSDG may not help convergence in the early stage of training. Shin et al. [31] proposed XorMixFL, a privacy-preserving XOR-based mixup data augmentation technique. The core idea of XorMixFL is collecting other clients' encoded samples, which are decoded only with the data samples of each client. Since both encoding and decoding are bit-wise XOR operations that intentionally distort raw data, data privacy is preserved to some degree. Similarly, Yoon et al. [32] proposed FedMix, which appropriately modifies Mixup [33], a popular data augmentation technique generating synthetic data by linear interpolation between samples, under the restrictions of federated learning. In FedMix, clients exchange averaged data comprised of M local samples. Then, FedMix modifies the local loss function to approximate the global Mixup with only averaged data, where global Mixup means that raw data are exchanged and directly used for Mixup between clients. However, the works indirectly sharing data information may cause some privacy issues. There is the possibility that individual data could be inferred from the exchanged information, such as batch normalization layers, encoded data, and averaged data. In addition, the ownership or data distribution of local datasets also could be inferred if the exchanged information includes client-specific information or labels. Therefore, with users' different privacy requirements, the above methods may not be directly applied in FL.

Moreover, Generative Adversarial Network (GAN) [34] has also been applied in FL for data augmentation. Jeong et al. [35] proposed FAug, which trains a conditional GAN [36] with the seed data samples uploaded by clients. The trained GAN generator is distributed to each client, and each client can replenish the minority classes until reaching an IID dataset. With a similar idea of FAug, Yan et al. [37] proposed FAu, which trains a WGAN [38] at the server and lets clients replenish the lack of their local datasets with it. Recently, GAN has also been applied in FL for facilitating COVID-19 detection. Nguyen et al. [39] and Zhnaget al. [40] proposed FedGAN and FedDPGAN, respectively. To better protect privacy, both FedGAN and FedDPGAN

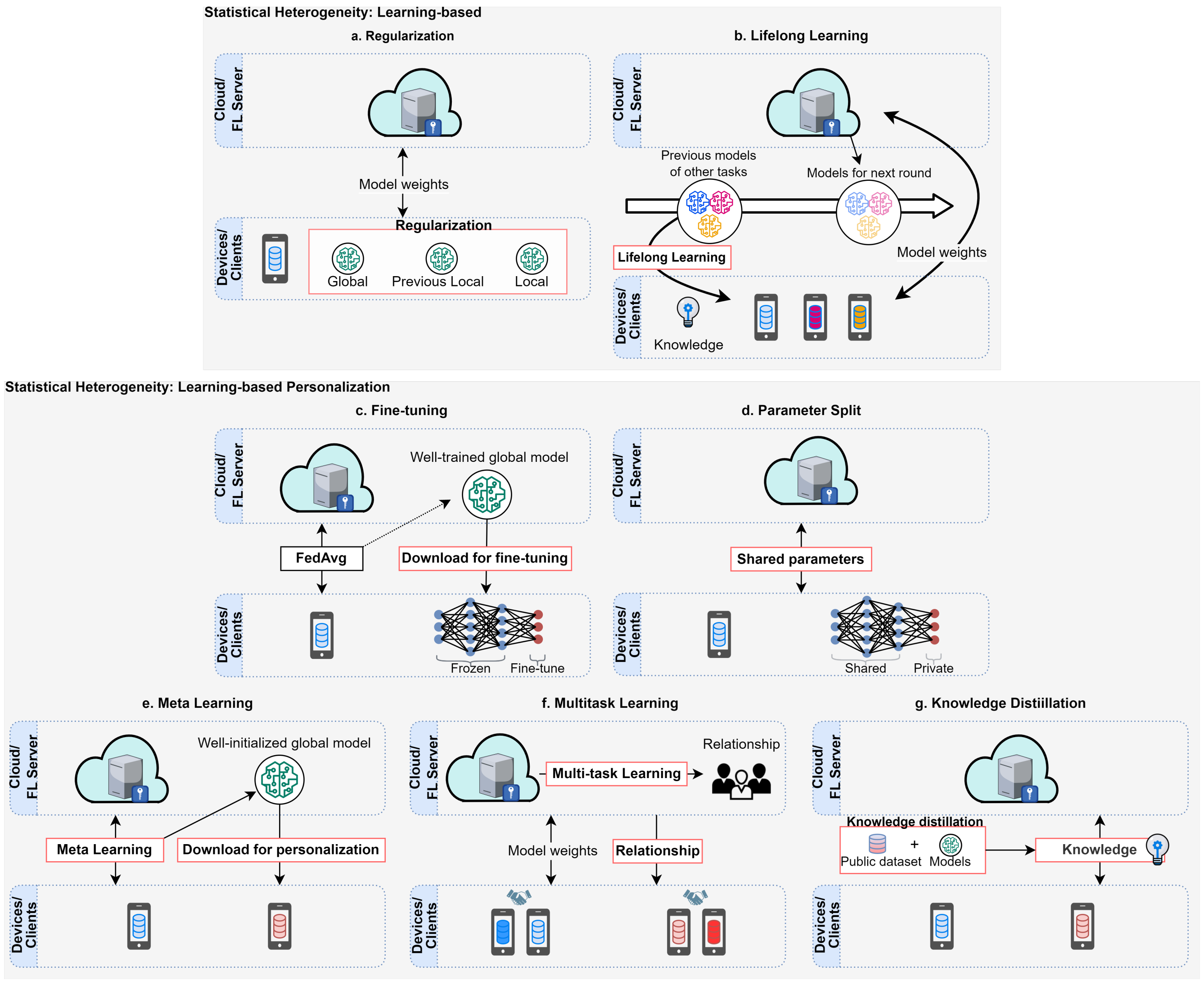

FIGURE 5: Illustration of learning-based approaches, including (a) regularization, (b) lifelong learning, (c) fine-tuning, (d) parameter split, (e) meta learning, (f) multitask learning, and (g) knowledge distillation.

train a GAN following the standard FL procedure instead of uploading raw data to the server like FAug and FAu. Specifically, the GAN model weights are exchanged between the server and medical institutions and updated iteratively on the client side. After the training, the GAN model is used to perform COVID-19 image augmentation and solves the non-IID issue. Furthermore, the differential privacy (DP) technique, which adds Gaussian noise in the training gradient or model weight, is integrated into FedGAN and FedDPGAN to alleviate the privacy leakage from the GAN model weight. However, although DP effectively increases privacy, it reduces the quality of data generated by GAN, which in turn degrades model performance. Achieving a balance between privacy preservation and model performance is still an open problem for further investigation.

B. LEARNING-BASED ALGORITHMS

On the non-IID data, the model weights of local models usually diverge, and such conflicting model updates further hurt the performance of the global model. To handle the weight divergence, learning-based algorithms modify the optimization objective or learning algorithms. More precisely, some algorithms add a regularization term into the loss function to prevent weight divergence. In addition, several existing DL techniques are applied to FL to help local or global models generalize better. Furthermore, the concept of personalization is proposed to train a specific local model for each client instead of a single global model. In this subsection, we review the ideas and techniques applied by the learning-based algorithms in detail.

1) Regularization

Regularization is a technique that lowers the complexity of a neural network and thus prevents overfitting and improves generalization while training DL models. In FL, regularization can be used to prevent local models from drifting away from the global optimal point and overfitting to their local objectives. Specifically, instead of only minimizing the local loss function $F_k(\omega)$, clients minimize the following objective, which considers the regularization terms $L_{reg}(\omega)$:

$$\min_{\omega} h_k = F_k(\omega) + L_{reg}(\omega) \tag{1}$$

In FL, several existing works mitigate weight divergence by introducing a variety of regularization terms (or proximal terms) to restrain local model updates to the global model. Generally, the regularization terms are composed of the local model, local historical models, or the global model, as shown in Fig. 5(a). Karimireddy et al. [41] proposed SCAFFOLD, which uses variance reduction to correct the weight divergence in clients' local updates. SCAFFOLD estimates the update directions for the global model (c) and local models (c_i). Then the difference of directions ($c - c_i$) is used as the correction terms to correct the local update and ensure the updates move towards the true optimum. Li et al. [7] proposed FedProx, which introduces a proximal term $\frac{\mu}{2}\|w - w_t\|$ to penalize local updates far from the global model. Furthermore, FedProx considers the interaction between statistical and system heterogeneity and allows for safely incorporating local updates with variable local epochs. Recently, Acar et al. [42] has shown that minimizing local empirical loss is fundamentally inconsistent with minimizing the global one under non-IID data. Motivated by the idea, FedDyn [42], which dynamically modifies the local objective with a proximal term, was proposed to ensure that the local optimal point is asymptotically consistent with the stationary point of the global empirical loss. In addition, with the intuitive insight that the global model integrating the knowledge of all local datasets can perform better than each local model, Li et al. proposed MOON [43], which utilizes the similarity between model representations to correct the local objective. Specifically, MOON adds a regularization term to minimize the distance between the representation of the global model and the local model and increase the distance between that of the current and previous local models. Although MOON achieves state-of-the-art results in learning visual representation, it incurs $\sim$ 3x cost in both memory and computation since calculating the loss function requires storing three models in memory and forward passing of each model every local epoch. In a resource-scarce FL setting, the resource requirement of MOON may not be satisfied.

Although the works [7], [41]–[43] have brought significant progress on the problem of non-IID data, as shown in [44], since the proximal terms restrain the weight divergence, the local convergence potential of clients is limited, and less information is aggregated per training round. Consequently, several above works do not provide stable performance across variable non-IID settings. To tackle the issue, Mendieta et al. proposed FedAlign [44], which uses a distillation-based regularization, rather than proximal restriction, to promote local learning generality. To be more specific, FedAlign regularizes the Lipschitz constant of the final block in the model for its representations. Furthermore, compared with MOON [43], FedAlign keeps memory and computation resource requirements to a minimum by focusing on the last block only.

2) Lifelong Learning

In lifelong learning, the challenge is to learn multiple tasks with the same model sequentially but without forgetting while learning the new task, where forgetting means hurting the performance of previous tasks. As shown in Fig. 5(b), under the statistical heterogeneity, it is intuitive to view learning a model on each local dataset as a separate learning task. Consequently, lifelong learning shares a common target with FL: learning a task without disturbing the others learned on the same model. With the above intuition, several works are proposed to apply lifelong learning to FL to solve statistical heterogeneity. In addition, the most common lifelong learning algorithm applied in FL is Elastic Weight Consolidation (EWC) [45], which aims to prevent catastrophic forgetting when moving from learning task A to learning task B. The idea of EWC is to identify the critical parameters in the model that are the most informative for task A. Then the model will be penalized for changing these parameters while training task B.

Kopparapu et al. [46] proposed FedFMC, a method that dynamically groups clients with similar archetypes together and regards each group as a task to learn. At first, FedFMC dynamically groups the clients with similar data distributions and makes each group trains its model separately with FedAvg. After the first step, several global models that learn different data distributions are obtained. Next, the merging process begins with a group's model. Then, it iteratively merges each group's model into the globally shared model with the EWC. In the end, FedFMC gets a global model that learns all different data distributions. In comparison with FedFMC, which treats groups as different tasks, FedCurv, proposed by Shoham et al. [47], treats each client as a different task. Specifically, FedCurv uses the EWC algorithm to identify important parameters of each client and then broadcast such information to selected clients in every round. Then selected client will be penalized for drifting away from the important parameters of others while performing its local training. However, the cost of maintaining all the historical information required by FEDFMC or FedCurv is expensive with the aspect of memory and communication, even with some optimization tricks. Motivated by considerable overheads brought by treating groups or clients as different tasks, Yao et al. [48] proposed FedCL, which treats learning on the server as a task and constrains local training with parameter information. Specifically, FedCL uses the EWC algorithm to evaluate important parameters of the global model on a proxy dataset on the server. Then the importance of parameters is transferred to the clients and is used to constrain the local training and alleviate the weight divergence. Since FedCL performs less EWC and requires less historical storage, it only incurs a few extra costs.

3) Personalization

In the vanilla FL setting, a single global model is trained to fit all clients. However, the global model generalizes poorly due to statistical heterogeneity. Therefore, training a single model may be insufficient for the FL environment, which often has non-IID data. Motivated by the observation, personalized FL (PFL) is proposed to provide a personalized model tailored to each client and better fit the local dataset. Generally, the personalization strategies are related to transfer learning (TL) [49], which extracts knowledge and experience from one or more source domains and applies to the target domain. Specifically, the local datasets are regarded as different domains, and the challenge of PFL is integrating the knowledge of all source domains and applying it to the personalized model (target domain) of each client under the FL setting. In this subsection, we discuss the key ideas and techniques in personalization strategies.

a: Fine-Tuning

Fine-tuning in DL means training a model from a pre-trained one instead of scratch. As illustrated in Fig. 5(c), the works that apply fine-tuning into FL involve two steps: 1) training a single global model via FL; 2) each client fine-tunes the global model with its local dataset. FedHealth [50] and FedSted [51] both follow the steps mentioned above to achieve personalization. In addition, several techniques, such as parameter freezing and correlation alignment (CORAL) layer, are introduced for better transferring the knowledge. Although the performance of personalized fine-tuning is highly dependent on the global model trained in the first step, fine-tuning algorithms can be easily compatible with FL algorithms optimizing the global model by adding a tuning step.

b: Parameter Split

As shown in Fig. 5(d), parameter split means that the model parameters are divided into shared and private parameters. Shared parameters are exchanged and updated between the server and clients, just like the normal FL process. In contrast, the private parameters of each client are updated separately and not shared with the server. The private parameters excluded from the averaging procedure can help achieve personalization and combat the negative effects of statistical heterogeneity. Bui et al. [52] proposed FURL, which treats user embeddings in a bidirectional LSTM architecture as private parameters. Furthermore, FURL theoretically and empirically shows that parameter split improves model performance in the FL setting. Arivazhagan et al. [53] proposed FedPer, a base + personalized layers approach for PFL. In FedPer, the last fully connected layer is set as a personalized layer to learn the domain-specific features. On the other hand, the FL process updates the base layers to learn low-level features that generalize all clients. With the similar idea of FedPer, FedRep [54] leverages all clients to learn global low-dimensional features and enables each client to maintain a private, personalized classifier. In contrast to FedRep, LG-FedAvg [55] divides parameters with opposite insights. Specifically, the private and shared parameters in LG-FedAvg are designed to be complementary: private parameters are responsible for capturing important features of local data for prediction. By contrast, shared parameters operate only on low-dimensional features and ensure that data from all clients can be classified correctly. In the above works, parameter split also shows additional advantages with respect to privacy and costs. Since the private parameters do not be transferred back to the server, the privacy-sensitive information in such parameters is kept local. In addition, communication cost is reduced because only shared parameters are required to be exchanged during training.

c: Meta Learning

Meta-learning [56] is a learning paradigm that learns "how to learn". In meta-learning, the goal is to let a model learn prior knowledge on a collection of tasks such that it can adapt faster and better when facing new tasks. The characteristic of meta-learning makes it particularly well-suited for PFL. The key insight applying meta-learning into FL is similar to that of lifelong learning: learning on each local dataset is viewed as a learning task. The common meta-learning applied in FL is model-agnostic meta-learning (MAML) [57], which can be applied to any gradient-based approach. Noteworthy, although the algorithm flow of federated meta-learning (Fig. 5(e)) seems like personalized fine-tuning (Fig. 5(c)), which trains a single global model and then performs personalization on it, the global models they trained have different meanings and targets. The global model trained by meta-learning optimizes the performance after adaptation to tasks, while the global model trained by personalized fine-tuning optimizes the performance before adaptation to tasks.

Jiang et al. [58] point out the connection between two widely used FL and MAML algorithms and discover that FedAvg can be interpreted as meta-learning when all clients possess the equal size of the local data. In addition, the authors proposed a fine-tuning stage using Reptile [59] to improve the initial model initialized by FedAvg. Given the relationship between metal learning and FL, this work shows that meta-learning is a critical technique that helps FL yield a better personalization result.

Chen et al. [60] proposed FedMeta, which aims to train an initialization for the global model using all clients. In FedMeta, the local training process is replaced by MAML or MetaSGD [61]. After the training, the well-initialized model is personalized with the local dataset by each client. With a similar idea, Fallah et al. [62] proposed Per-FedAvg, a variant of FedAvg based on the MAML formulation. In Per-FedAvg, the authors evaluated two forms of approximations to avoid computing second-order gradients, which is computationally expensive and required by MAML. Furthermore, they also characterize how this performance is affected by the closeness of underlying distributions of user data, measured in terms of Total Variation and 1-Wasserstein metric. Compared with the works [61], [62] attempting to approximate Hessian matrix, pFedMe [63], a federated meta-learning formulation using Moreau envelopes, only needs gradient calculation using a first-order approach. Specifically, pFedMe uses Moreau envelopes as clients' regularized loss functions, decoupling personalized models' optimization from learning the global model. Although the problem formulation of pFedMe has a similar meaning to Per-FedAvg, pFedMe pursues the performance of personalized and global models in parallel instead of using the global model as the initialization.

Different from the works [58], [60], [62], [63] optimizing the personalized models for individual clients, GPAL [64] aims to utilize meta-learning to prepare an initial model that can quickly adapt to any group of clients to achieve few-round FL. In more detail, the goal of GPAL is to prepare an initial model that fits a group of clients with new tasks within only a few "rounds", while personalized FL aims for an initial model that leads to personalized models within a few "local updates". Although these are two completely different problems and solutions, they both make the initial model serve as prior knowledge, preventing local models from overfitting the local data and ensuring fast convergence.

d: Multitask Learning

Multitask learning (MTL) is an approach that shares domain-specific information between related tasks to make the model better generalize on the target task. As illustrated in Fig. 5(f), MTL aims to model the relationships between tasks and learn models for multiple related tasks. By regarding the clients in FL as different tasks, MTL approaches can directly capture relationships among non-IID datasets, which helps clients with similar data enable each other to generalize the model better. Furthermore, since each task might not be closely related to all tasks, simply aggregating the weights with unrelated tasks may hurt performance. Therefore, applying MTL to FL also helps clients neglect the information from unrelated clients and further alleviates weight divergence while aggregating model weights.

Smith et al. [65] proposed MOCHA, which extends distributed MTL into FL. MOCHA models the pair-wise collaboration relationships between clients by a bi-convex alternating method that performs well on convex cases. However, MOCHA is not applicable to non-convex problems due to its requirement of strong duality. This motivated Corinzia et al. [66] to propose VIRTUAL, an algorithm for federated multitask learning for general non-convex models. In VIRTUAL, the network between the server and clients is treated as a star-shaped Bayesian network, and learning is performed on the network using approximated variational inference.

Li et al. [67] proposed Ditto, a simple MTL framework addressing the competing constraints of accuracy, fairness, and robustness in FL. Ditto only considers two tasks: the global model and the local model. To relate the two tasks, Ditto incorporates a regularization term, as same as that in FedProx [7], which encourages the personalized models to be close to the optimal global model. Huang et al. [68] proposed FedAMP, an algorithm employing federated attentive message passing to facilitate similar clients to collaborate more. Specifically, FedAMP identifies the relationships between clients by the similarity of clients' model weights. If the model weights of the two clients are similar, they contribute more to each other's local training. Compared with capturing relationships by the similarity of model weights, SFL [69] achieves it by leveraging the graph-based structural information among clients. To be specific, the client-centric model aggregation will be conducted along the relation graph's structure built by a graph convolutional network [70].

However, a general limitation of the above works is that the MTL algorithm is justified qualitatively but not on the basis of clear statistical assumptions on local data distributions. To tackle the limitation, Marfoq [71] proposed FedEM, which studies federated MTL under the flexible assumption that each local data distribution is a mixture of M underlying distributions. In FedEM, a personalized model is a linear combination of M shared component models. All clients are required to learn the M components jointly, while each client learns its personalized one. Although FedEM requires M times of computation and communication costs than FedAvg, the extensive experiments show that FedEM achieves state-of-the-art model performance on accuracy, fairness, and generalization.

e: Knowledge Distillation

Knowledge distillation (KD), referred to as a teacher-student paradigm, is a technique used to transmit learned information in a model-agnostic way. By encouraging the student model to approximate the output logits of the teacher model on the shared data, the student can have a similar or better performance with the teacher. As shown in Fig. 5(g), with the help of a public dataset and knowledge distillation algorithms, we could extract knowledge of local models or the global model and share it to help local training or global model updates. Besides, in the original framework of FL, all clients are required to perform training on a particular model architecture. However, in real-world scenarios, each client might expect to design its model independently due to the differences in personalized requirements, data distribution, and business purposes. Therefore, it poses a practical challenge: model heterogeneity. Preceding weight-based aggregation methods are developed under the assumption that local models share the same model architecture, which can not work on heterogeneous models. To achieve PFL with model heterogeneity, KD is introduced into FL as a translation protocol enabling clients to share knowledge in a model-agnostic way. Moreover, KD alleviates the weight divergence issue since it does not follow weight-based aggregation, which is shown to be more effective than simple FedAvg.

In general, the main difference of distillation-based FL algorithms is the direction of distillation, which can be classified into three types: 1) distillation between clients, 2) distillation of knowledge from clients to the server to learn a stronger global model, and 3) distillation of knowledge from the server to each client to learn a better-personalized model.

For the first type of direction of distillation, the knowledge communication is mainly based on a public dataset. Jeong et al. [35] proposed federated distillation (FD), which exchanges not the model weights but the model output in the FL process. In FD, each device treats itself as a student and sees the mean model output of all the other devices on a public dataset as its teacher's output. Similarly, Li et al. [72] proposed FedMD, which re-purposes the public dataset as the basis of knowledge transfer. In a round of FedMD, the server collects all clients' model output computed on a public dataset as the consensus. Each client then learns to approach the consensus on the public dataset and fit its own private dataset iteratively for personalization. Huang et al. [73] proposed FCCL, which constructs a cross-correlation matrix to learn a generalizable representation and utilizes KD to handle catastrophic forgetting. Specifically, FCCL distills the knowledge of the local models learned in previous rounds and the initially pre-trained local model to avoid forgetting inter and intra-domain information, respectively. Despite the high algorithm complexity, FCCL achieves state-of-the-art model performance in the comprehensive evaluation.

However, the prerequisite, preparing a public dataset, can leave the above works [35], [72], [73] infeasible for many scenarios. A carefully engineered dataset may only sometimes be publicly available on the server. To tackle such limitation, Zhu et al. [74] proposed FedGen, a data-free KD approach that learns a generative model by ensembling the knowledge of multiple local models over the latent space. To be more specific, given a target label, the generative model can yield feature representations consistent with the ensemble of client predictions. Then this generator is broadcasted to clients to help with local training tasks by augmented samples that embody the distilled knowledge from other clients.

Besides distillation between clients, some works focus on distilling knowledge from clients to the server for a better global model. For example, Lin et al. [75] proposed FedDF, which investigates more powerful and flexible aggregation schemes for FL. In FedDF, FedAvg is first performed among clients to build an initial student model. Then all local models are treated as teacher models to transfer knowledge to the student model via ensemble distillation. Moreover, knowledge may be distilled in a bidirectional manner between the clients and the server. He et al. [76] proposed FedGKT, a variant of the alternating minimization approach to train small local models and periodically exchange their knowledge by KD with a large server-side model. Specifically, the sever-side model takes the feature extracted from the local datasets by local models as inputs and uses the soft labels uploaded by clients to distill knowledge from local models. Similarly, the local models utilize the soft labels predicted by the server-side model to absorb its knowledge. Besides using KD to improve both local and server-side models, another principal advantage of FedGKT is reducing the burden on the computation of clients by shifting it to a more powerful server. However, FedGKT makes clients share ground truth labels with the server, which causes potential privacy risks.

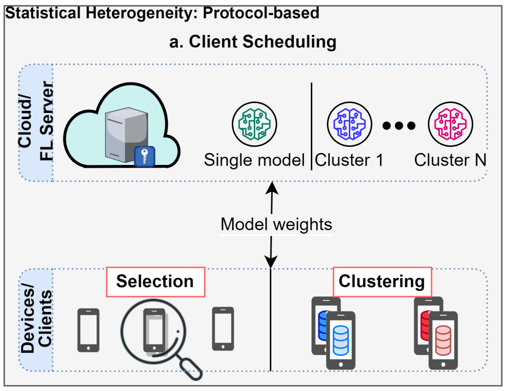

FIGURE 6: Illustration of protocol-based approaches, including (a) client scheduling.

C. PROTOCOL-BASED ALGORITHMS

Protocol-based algorithms aim to design a system architecture or a communication protocol to help counterbalance the bias between clients introduced by non-IID data. Intuitively, an implicit connection exists between model weights and the data distribution of the training dataset. Therefore, actively arranging clients' participation will reduce the negative impact of weight divergence. In this subsection, we detail the concept above and explore the architectures and protocols proposed by the protocol-based algorithms.

1) Client Scheduling

In FL, it is impractical to perform local training on all clients simultaneously due to the limited bandwidth and unstable network connectivity. Therefore, FL algorithms usually select a small subset of clients to participate in each round, which can exacerbate the adverse effects of statistical heterogeneity. In addition, randomly selecting the clients may not be optimal. For example, as shown by Cho et al. [77], the model convergence can be sharply accelerated by biasing the client selection towards clients with higher local losses. In addition, Wang et al. [78] proposed FAVOR, which uses reinforcement learning (RL) to learn which clients have a higher priority in participating in training. FAVOR models the per-round FL process as a Markov Decision Process with the states represented by the model weights, the actions represented by client selection strategies, and the rewards represented by the function of test accuracy achieved by the global model so far. By carefully designing the states and reward functions, the RL agent can learn the potential interplay between client selection strategies and the model performance and further take the strategy, which aggressively improves the accuracy of the global model. Besides the client selection, Astraea, proposed by Duan et al. [22], alleviates the bias of non-IID data by the mediator-based multi-client rescheduling. Specifically, Astraea divides the clients based on their data distributions and makes the distribution of each group close to uniform. Then the clients in the same group perform local training sequentially to counterbalance the bias introduced by the non-IID data of the clients.

Moreover, some client scheduling algorithms introduce the concept of personalization mentioned before. As shown in Fig. 6, each client can arrive at a more specialized model by clustering the homogeneous clients into the same cluster and training a shared model for each cluster. Next, we introduce the algorithms focusing on clustering for personalization. Sattler et al. [79] proposed CFL, which is viewed as a post-processing method achieving greater or equal performance than conventional FL. With the theoretical finding that the cosine similarity between the weight updates of clients is highly indicative of the similarity of their data distributions, a cosine similarity-based bi-partitioning algorithm is used to divide the clients into clusters. However, the recursive bi-partitioning algorithm requires multiple training rounds to separate all heterogeneous clients. The above drawback limits the scalability of CFL due to the high computation and communication costs. Compared with CFL, which performs clustering every round, FL+HC, proposed by Briggs et al. [80], reduces clustering into a single step to reduce costs. Specifically, the clustering step begins after several training rounds of the normal FL process. At first, the global model is broadcasted and fine-tuned by each client. Then, the difference between the global and fine-tuned local models is used to judge the similarity between clients. Finally, each cluster's FL training is performed independently to generate multiple personalized models. The operation of FL+HC is computationally intensive when there are many clients. Although the operation occurs on the server, the costs of FL+HC still need to be considered while deployed in a real-world environment.

Different from CFL and FL+HC, which do not require setting the number of clusters while clustering, some other algorithms need. Xie et al. [81] proposed FeSEM, which learns multiple global models and simultaneously derives the optimal matching between clients and clusters. FedSEM proposed a formulation to minimize the distance between each local model and the global model of its cluster. Then expectation-Maximization [82] with an additional step, i.e., updating the local model, is applied to solve the distance-based objective function of clustering. FedSEM is an iterative procedure that updates cluster assignments each iteration. The iteration is repeated until convergence. Ghose et al. [83]proposed IFCA, an iterative federated clustering algorithm. Given the number of clusters k, the server constructs k global models and broadcasts them to all clients. Then each client estimates its cluster identity by finding the global model with the lowest local loss. After all clients send their cluster identities and gradients back to the server, the server collects all the updates from the clients whose cluster identities are the same and updates the model of the corresponding cluster. However, the communication cost of the broadcasting step of IFCA is k times as FedAvg. In addition, the edge devices may not have enough resources to store k models in the memory or storage and further be excluded from the training.

D. SUMMARY AND LESSON LEARNED

In this section, we have discussed data-based, learning-based, and protocol-based approaches for statistical heterogeneity in FL. We summarized and compared the works in terms of their key ideas, disadvantages, and limitations, as shown in Table. 2.

Data-based approaches aim to reduce the degree of non-IID by sharing raw data or generating synthetic data. Although data sharing or data augmentation is easy to implement in the realistic FL system and is efficient in relieving accuracy reduction, their applicability is limited due to the possibility of privacy leakage and the potential communication cost of transmitting data information. In data sharing, the server only sometimes has an available globally shared dataset, and the clients may not be willing to share any private data. Therefore, the data sharing approaches may only be applicable in some FL scenarios. On the other hand, although the data augmentation methods avoid sharing raw data by generating synthetic data, they bring additional costs and a trade-off between privacy and model performance. For example, the encode-decode operations and exchanging process in the [31], [32] brings overhead proportional to the number of synthetic data. In addition, training GAN models used to augment local datasets [35], [37], [39], [40] exposes a heavy burden to the clients with limited resources. For the trade-off, performance improvement depends on the quality of synthetic data, but the higher the quality of synthetic data, the higher the possibility of privacy leaks. How to tackle the trade-off and achieve privacy preservation and performance improvement simultaneously is needed to be clarified in the future.

Learning-based approaches aim to tackle statistical heterogeneity by modifying learning algorithms or applying existing DL techniques. One of the key insights of learning-based approaches is that weight divergence can be relieved by punishing local models whose update direction or feature representation is far from that of the global model [7], [41]–[43]. To achieve it, different varieties of regularization terms are introduced to the local loss function to quantify the bias of local updates. Similarly, lifelong learning achieves the same purpose by identifying the important parameters of the local or global models and using them to regularize local training [46]–[48]. Another key insight of learning-based approaches is personalization: providing a personalized model better fits the local data for each client instead of a single global model. The fundamental challenge of personalization is how to identify and transfer helpful knowledge to build personalized models. Therefore, most PFL algorithms are related to transfer learning. Among the approaches, fine-tuning and parameter split are easy to implement. However, they rely on FedAvg to aggregate the knowledge of all clients, so their performance highly depends on that of the global model obtained from FedAvg and still suffers from the non-IID data. Similarly, the performance of meta-learning methods also depends on the initial global model, which learns the prior knowledge of all clients. In addition, the meta-learning algorithms theoretically require computing Hessian terms, which are computationally prohibitive. Although some approximation methods are proposed to reduce its computation overhead, the costs of meta-learning methods are still considerable. Multi-task learning excels in modeling the relationships between clients and enables clients to learn better with the help of related clients. However, the cost of modeling the relationship between such a significant amount of clients may be infeasible in the FL environment. Compared with the above methods, KD methods give each client better flexibility in designing its personalized model architectures. In addition, KD methods also relieve weight divergence by utilizing KD to transfer knowledge instead of model weight aggregation. Nevertheless, most KD methods require a representative proxy dataset, which may be unavailable in the FL setting. Moreover, KD methods' privacy analysis and protection mechanism should be discussed more.

Protocol-based approaches aim to design a system architecture or a communication protocol to help relieve the bias introduced by non-IID data. The existing works [77] have shown that actively selecting clients each round is a better strategy than random selection. Based on the above idea, several ideas about client scheduling have been proposed and efficiently improve model performance in FL. However, most client scheduling algorithms do not consider system heterogeneity, and not all clients are always ready for training. Therefore, the training efficiency may be degraded due to biased client selection. In addition, client scheduling methods usually assume that all clients' statuses, such as local data distribution and their separate model weights, are known. However, it is infeasible in the FL setting. Moreover, scheduling such a large amount of clients places an enormous burden on the server.

IV. TAXONOMY OF SYSTEM HETEROGENEITY

In this section, we present a unique taxonomy of works on system heterogeneity according to their key insight and technique. Based on our proposed taxonomy, the works are divided into *learning-based* and *protocol-based*. The insight of taxonomy for system heterogeneity is consistent with the one for statistical heterogeneity. Learning-based algorithms aim to solve system heterogeneity by revising the learning algorithm or the information of model updates. In addition, protocol-based algorithm solve it by modifying the FL system architecture or communication protocols. Next, we further introduce and discuss the algorithms in detail.

A. LEARNING-BASED ALGORITHMS

Under system heterogeneity, FL algorithms suffer from stragglers. To solve the straggler issue, learning-based algorithms modify the learning algorithm to reduce the impact of stragglers or balance the workloads. More specifically, some algorithms update the global model with an asynchronous scheduler to tolerate slow clients. In addition, workload balancing re-balances the workload of clients to accelerate the training of stragglers. For example, some algorithms enable heterogeneous local epochs, model architecture, or compression rate of model updates. In this subsection, we explore the learning-based algorithms' main ideas and challenges.

1) Asynchronous Training

Asynchronous training in distributed learning and FL means that the global model can be updated without waiting for others clients. As illustrated in Fig. 7(a), by introducing asynchronous algorithms into FL, each client can train and update the global model at its own pace, and the stragglers do not slow down the whole training process. However, asynchronous algorithms face the challenges of staleness, which means the out-of-date updates from clients, especially slow-responding clients, may not provide helpful information for training or even hurt the global model performance.

To address the staleness, several asynchronous approaches correct the model updates of clients according to staleness. Xie et al. [84] proposed FedAsync, an asynchronous federated optimization algorithm that adaptively controls the trade-off between the convergence rate and variance reduction based on the staleness. Specifically, the model update uploaded to the server is re-weighted by a function of staleness and then aggregated to the global model. The staler the model update is, the smaller it will be re-weighted to have a minor effect on the global model. Lu et al. [85] proposed PAFLM, which contains a dual-weights correction to solve the performance degradation caused by the staleness issue. The dual-weights of PAFLM consider two factors, the proportion of the local samples to the total samples and the time difference between downloading the model and uploading the corresponding model update. The dual-weights correct uploaded model updates according to the above two factors, where the more samples a client has and the less stale it is, the more the model update will be retained. Otherwise, the value of the model update will be corrected to be smaller. Compared with the previous work [85], Chen et al. [86] has the opposite view. Chen et al. thinks that clients with high latency may have more data samples, so the clients uploading fewer updates need to be given greater step sizes to compensate for the loss.

Besides correcting the model updates, some works proposed semi-asynchronous algorithms which combine the advantages of both synchronous and asynchronous algorithms, such as tackling the stragglers and mitigating the impact of staleness. For example, Wu et al. [87] proposed SAFA, a semi-asynchronous training scheme that only synchronizes a specific subset of clients. SAFA classifies all clients into

Method		Key Ideas	Disadvantages and Limitations
Data-based	Data Sharing	Sharing raw data collected from clients or the server with clients to reduce the degree of non-IID.	• Sharing raw data causes critical privacy issues. • The communication cost of transmitting raw data may be intolerable. • The trade-off between privacy, utility, and model performance is still unclear.
	Data Augmentation	Generating synthetic data by traditional data augmentation or indirectly sharing data information for clients to reduce the degree of non-IID.	• Traditional data augmentation methods may not be feasible in highly skewed non-IID scenarios. • The process of generating synthetic data causes privacy issues and additional overheads. • The trade-off between privacy and model performance is still unclear.
Learning-based	Regularization	Introducing a regularization term to correct the update direction and prevent weight divergence.	• The regularization terms may slow the local convergence speed due to the weight restriction. • Calculating regularization terms brings additional computation and memory overheads.
	Lifelong Learning	Treating learning on different clients or the server as individual tasks and applying lifelong learning to learn tasks without forgetting.	• Performing EWC algorithm and exchanging weight information result in a heavy workload in aspects of computation and communication.
	Fine-Tuning	Let each client perform fine-tuning on the well-trained global model to obtain a better-personalized model.	• Performance is highly depended on the trained global model.
	Parameter Split	Splitting the model into shared and private parameters. Each client can personalize their model by training private parameters independently.	• It is difficult to determine the optimal splitting strategy for each scenario.
	Meta Learning	Utilizing meta-learning to obtain an initial model aggregating all clients' prior knowledge. Then clients train it separately to obtain a personalized model.	• Performance is highly depended on the initial global model. • The meta-learning algorithms, such as MAML, theoretically require computing Hessian terms, which are computationally prohibitive.
	Multitask Learning	Treating learning on different clients as individual tasks and applying multitask learning to model the relationships between tasks and make the model generalize better on the target task.	• Modeling the relationships between clients may be unaffordable due to its complexity. • Most works only justify algorithms qualitatively but not under the clear statistical assumption.
	Knowledge Distillation	Introducing KD as an alternative aggregation method to tackle weight divergence better, protect privacy, and achieve model heterogeneity.	• The algorithms often require a representative proxy dataset, and their performance depends highly on the dataset. • Lack of analysis and comparison of the degree of privacy leakages between gradient, model weights, and logits. • Data-free KD methods may violate the privacy regulation in FL.
Protocol-based	Client scheduling	By actively arranging client selection, we can counterbalance the bias introduced by non-IID data or even achieve personalization through clustering clients.	• Most works do not consider system heterogeneity; therefore, the training efficiency may decrease due to biased client selection. • The assumption that all clients' statuses are known is not feasible in the real world. • Scheduling amount of clients may bring considerable overhead to the server.

TABLE 2: Summary of techniques in *Taxonomy of Statistical Heterogeneity*

three states. The first is up-to-date clients that have completed the previous local training. The second is deprecated clients, which train on the global model that are too stale. The last one is tolerable clients, which trains on a global model, not the latest version but not stale either. SAFA only requires up-to-date and deprecated clients to stay synchronous with the server. Chai et al. [88] proposed FedAT, a FL algorithm with asynchronous tiers. FedAt logically partitions all clients into tiers based on their response latency, where the clients with similar latency are divided into the same tier. Then all tiers in FedAT participate in the training simultaneously, where intra-tier training is synchronous and cross-tier training is asynchronous. However, frequently interacting with the faster tiers relative to slower ones would introduce biases to the global model. To solve it, similar to the insight of Chen et al. [86], FedAT assigns higher coefficients to the slower tiers while performing aggregation so that the global model would not bias toward the faster tiers.

Although asynchronous methods can alleviate the straggler issue, aggregating individual model updates is incompatible with secure aggregation [89], which could leak privacy. To solve it, FedBuff is proposed by Nguyen et al. [90]. In FedBuff, clients begin and finish local training asynchronously. However, the aggregation is not performed immediately upon receiving model updates. Instead, model updates are stored in a buffer until K model updates are in the buffer so that FedBuff can achieve privacy against the honest-but-curious threat model through secure aggregation and

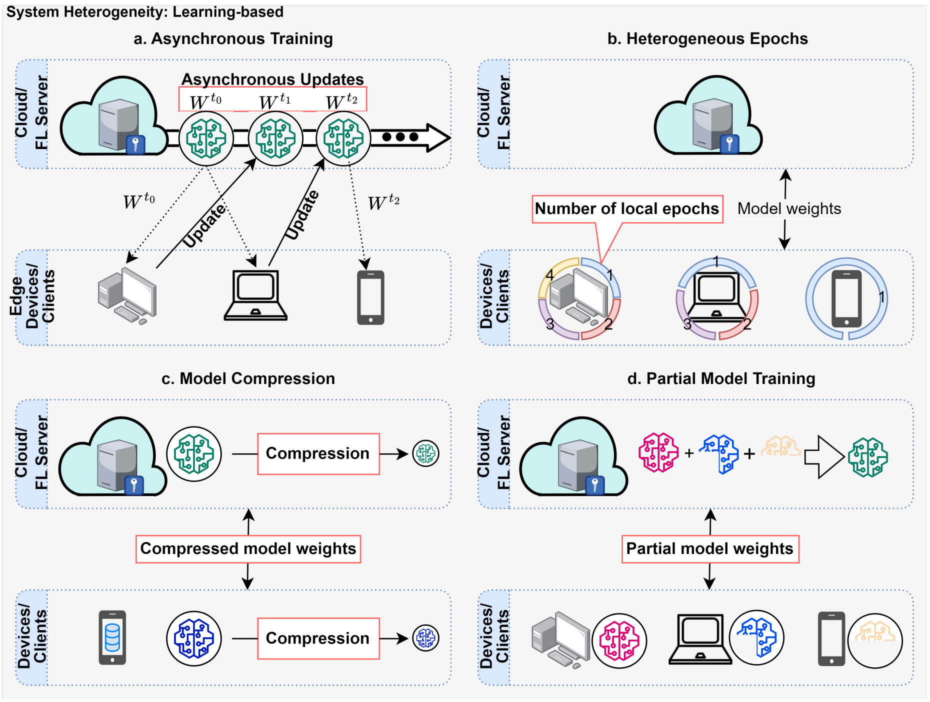

FIGURE 7: Illustration of learning-based approaches, including (a) synchronous training, (b) heterogeneous epochs, (c) model compression, and (d) partial model training

differential privacy [91]. Noteworthy, FedBuff is integrated into PAPAYA [92], the first production FL system to support asynchronous and synchronous training at scale.

2) Heterogeneous Epochs

In most FL algorithms, the number of local epochs is the same across all clients. However, with the various size of local datasets, computation, and communication speeds, the number of local epochs completed by clients within a given time interval can vary greatly. Therefore, making each client perform a uniform number of local epochs is unrealistic. Although tolerating partial works (heterogeneous local epochs) for solving system heterogeneity is effective (Fig. 7(b)), it may result in heterogeneity in the model progress at each client and further hurt the convergence speed and test accuracy of the global model.

To enable heterogeneous local epochs and solve the adverse impact brought by it, some solutions are proposed. Li et al. [7] proposed FedProx, which can be viewed as a generalization and re-parameterization of FedAvg. FedProx allows for variable amounts of local epochs to be performed across clients based on their system resources and then aggregate the partial work sent from the stragglers. To safely incorporate the partial work, FedProx adds a proximal term mentioned in Section. III-B1, which penalizes the model updates far from the global model. Different from the proximal term, Ruan et al. [93] alleviate the bias of heterogeneous local epochs by re-weighting the model updates. By assigning clients that run fewer local epochs a greater aggregation coefficient, the difference in training progress is compensated, and the bias introduced by tolerating partial works is reduced. Wang et al. [94] proposed FedNova, a normalized averaging method. Specifically, FedNova first normalizes the model updates and accumulates them. Next, the accumulative model updates are re-scaled before being aggregated into the global model. The normalization step can help counterbalance the different step sizes of model updates. In addition, the re-scaling step can ensure that the normalization step does not decrease the convergence speed.

3) Model Compression

Communication is often the bottleneck of FL, especially for the stragglers, due to the limited bandwidth and unstable network connection. Therefore, applying compression algorithms to reduce the communication cost (Fig. 7(c)) is a popular research topic. Although existing compression methods can efficiently reduce communication costs for distributed learning, there are several challenges to applying them to FL due to the unique characteristics of the FL environment. First, the compression methods may degrade the ability of the server to update the global model accurately and further hurt model performance. The trade-off between compression and model accuracy in the FL environment remains to be determined. Second, the compression method should be robust to features of the FL environment, such as non-IID data, small batch sizes, and partial client participation. In summary, how to design an efficient compression method to minimize communication cost and accuracy reduction in FL or tackle the trade-off between compression rate and model accuracy is the point we discuss here.

Recently, several works on compression methods have been proposed to meet the requirements of the FL environment. Sattler et al. [95] proposed STC, a new compression framework designed explicitly for FL. STC first shows that top-k sparsification suffers the least from non-IID data out of all compression methods. However, its utility is limited in the FL setting as it only directly compresses the upstream communication. Motivated by the observation, STC extends the existing top-k gradient sparsification compression technique with a novel mechanism to enable downstream compression, ternarization, and optimal Golomb encoding of the weight updates. Mills et al. [96] proposed CE-FedAvg, composed of distributed Adam optimization and compression methods. For the compression, CE-FedAvg comprises sparsification followed by quantization. To sparsify gradients, the top $(s-1)\%$ of deltas with the largest absolute value are chosen for each tensor. After sparsification, the weights are quantized from 32-bit floats to 8-bit unsigned ints with exponential quantization, which prevents gradient explosion caused by Adam. Shlezinger et al. [97] proposed an encoding-decoding strategy that mitigates the effect of quantization errors on the ability of the server to recover the updated model accurately. Specifically, the method encodes each model update with subtractive dithered lattice quantization. Furthermore, the theoretical analysis shows that this error can be bounded by a term that decays exponentially with the number of users. Xu et al. [98] proposed FTTQ, which optimizes the quantized networks on the clients through a self-learning quantization factor. With the observation that two quantization factors in standard TTQ will converge to the same absolute value, FTTQ adopts one quantization factor instead of two in each layer to reduce communication and computation costs. Additionally, a single quantization factor may also alleviate weight divergence. Haddadpour et al. [99] proposed FedCOMGATE, which lowers the communication overhead by periodic averaging and exchanging compressed messages. To be more specific, FedCOMGATE uses compressed messages for uplink communication to reduce communication costs. In addition, to make the compression method robust to non-IID data, FedCOMGATE applies local gradient tracking that ensures that each node uses an estimate of the global gradient direction to update its model locally. From the algorithmic standpoint, FedCOMGATE is similar to SCAFFOLD [41], which corrects the local update and ensures the updates move towards the true optimum, but FedCOMGATE is much simpler and does not require any extra control variable.

Most existing compression methods or the corresponding convergence analysis typically require identical compression rates across all the clients, which can limit their applicability and effectiveness. As mentioned, the trade-off between compression rate and model accuracy in the FL environment remains unclear. Dynamically adapting compression rate has an opportunity to minimize accuracy reduction brought by compression error. In addition, it is important to take the heterogeneity in clients' communication capacity and energy consumption into account to further improve training efficiency and robustness in the real environment. Han et al. [100] proposed FAB-top-k, a fairness-aware online learning method that ensures that different clients provide a similar amount of updates. Specifically, with the goal of minimizing the overall training time, FAB-top-k uses an estimated derivative sign and adjustable search interval to determine the optimal value of gradient sparsity for each client. Moreover, by replacing training time with another type of additive resource, FAB-top-k can be easily extended to minimize other resource consumption. Li et al. [101] proposed FT-LSGD-DB, which balances the energy consumption between local computation and communication from the long-term learning perspective. Specifically, FT-LSGD-DB allows the client to perform gradient sparsification with different degrees by using generalized Benders decomposition and inner convex approximation to find a satisfactory solution for compression rates. In addition, FT-LSGD-DB novelly incorporates error compensation and batch size increment into FL procedures to accelerate model convergence. From the theoretical respect, Cui et al. [102] for the first time systematically examine the trade-off identifying the influence of the compression error on the final model accuracy with respect to the learning rate by factoring the compression error into the convergence rate analysis. Then, based on the derived convergence rate, the authors establish the policy to maximize the final model accuracy by adapting compression rates in accordance with learning rates.

4) Partial Model Training

The basic assumption of FedAvg is that all local models have to share the same architecture as the global model. However, this assumption makes the model architecture complexity limited by the most indigent client. Furthermore, assigning a uniform workload to the clients with various computation and communication capabilities can cause the straggler issue. Therefore, as shown in Fig. 7(d), the objective of partial model training is to achieve model heterogeneity and assign heterogeneous models with different complexity levels according to clients' capabilities adaptively. Although partial model training shares the same objective with KD, a model-agnostic algorithm mentioned in Section. III-B3e, the two methods are quite distinct from each other. First of all, partial model training aims to enable clients only to perform training on and exchange a subset of the model but still generate a single global model, while KD allows clients to perform training on completely different models and achieve personalization. Secondly, partial model training follows a weight-based aggregation method as in FedAvg, instead of using KD as a translation protocol and exchanging knowledge by a public dataset. Recently, the main research direction of works about partial model training is how to generate optimal subnetworks from the global model for each client, with respect to size and component.

Diao et al. [103] proposed HeteroFL, which proposes the concept of model heterogeneity and identifies its effectiveness. To generate the subnetworks, HeteroFL varies the width of hidden channels and ensures that the local and global model architectures are within the same model class, which stabilizes global model aggregation. Moreover, the global model in HeteroFL is constructed from the union of all disjoint sets of the partition. Sidahmed et al. [104] proposed FedPT, which splits the model into trainable and non-trainable. In FedPT, only the trainable parameters are exchanged and updated by the clients to reduce costs. However, FedPT highly depends on the network architecture, and it may be hard to find the optimal partition of parameters. Similar methods are also evaluated on neural networks for speech recognition [105], [106]; the experiment results show that different model architectures have different abilities to handle aggressive freezing rates. Chen et al. [107] proposed APF, which adaptively freezes (fixes) a subset of model parameters. Specifically, APF freezes parameters in intermittent periods, and the freezing periods are tentatively adjusted in an additively-increase and multiplicatively-decrease manner, depending on if the previously frozen parameters remain stable in subsequent iterations.

On the other hand, some works take inspiration from dropout [108] or pruning [109] to implement partial model training. In the traditional dropout, the parameters are multiplied by a random binary mask to drop a fraction of the parameters during each training pass. Empirically, traditional dropout is used as a regularization strategy for solving overfitting [110], but this technique is used for system-level concerns here. In addition to dropout, traditional pruning reduces the parameter counts to reduce the complexity of the model. The important research on model pruning is the "lottery ticket hypothesis [111]", which claims that an optimal subnetwork can be identified through pruning, and such subnetwork can reach test accuracy comparable to the original network. Compared with the works mentioned before [104]–[107], the works motivated by dropout or pruning pay more attention to how to select the best subnetwork for the global model or each client.

Caldas et al. [112] proposed Federated Dropout, which efficiently trains on a smaller subset of the global model. To realize communication and computation savings, Federated Dropout zeros out a fixed number of activations at each fully-connected layer and drop out a fixed percentage of filters. Then the sparse model is re-packed as a smaller dense model. Xu et al. [113] proposed ELFISH, a resource-aware FL framework. ELFISH first identifies the number of kept parameters for each client based on its resource constraints. Next, ELFISH selects the parameters with larger updates last round since larger parameter updates imply more significant impacts on the global model. Horvath et al. [114] proposed FjORD, which introduces ordered dropout into FL to enable partial model training. Ordered dropout is a mechanism ordering knowledge representation in nested subnetworks of the original network. In addition, ordered dropout is shown that it can recover the singular value decomposition in the case where there exists a linear mapping from features to response. Munir et al. [115] proposed FedPrune, which prunes the global model for slow clients based on resource characteristics. Specifically, FedPrune sorts parameters based on their post-activation values and picks the top m parameters that satisfy the model size constraint as the subnetwork. In addition, FedPrune uses insights from the central limit theorem to aggregate clients' model weights to better generalize the global model under non-IID data. Li et al. [116] proposed Hermes, a communication and inference-efficient FL framework. Hermes applies structured pruning to find a subnetwork for each client. Moreover, instead of taking the average over all parameters of clients, Hermes performs the average on only overlapped parameters across each subnetwork. Different from the works assigning a subnetwork for each client, PruneFL, proposed by Jiang et al. [117], adapts the global model size during the training process. Specifically, PruneFL first performs initial pruning on the global model at a selected client, and then the global model is pruned iteratively during the training process.

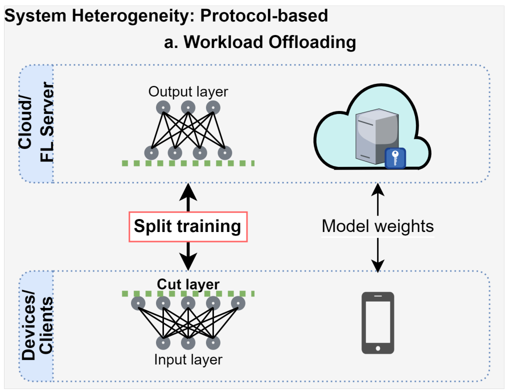

FIGURE 8: Illustration of protocol-based approaches, including (a) workload offloading

B. PROTOCOL-BASED ALGORITHMS

As mentioned, only a subset of clients is selected to participate in training at each round. The slowest straggler bounds the time cost of a round. To solve the straggler issue, protocol-based algorithms modify the client selection schemes or try to offload the stragglers' workload. Intuitively, limiting the time stragglers participate in training can help increase training efficiency. Motivated by it, some client scheduling algorithms that focus on actively selecting clients at each round are proposed. On the other hand, some algorithms bring the concept of workload balancing into FL. Specifically, the algorithms try to offload the stragglers' workload to the server and even other clients so that the completion times of the selected clients are close. In this subsection, we will detail the algorithms for client scheduling and workload offloading.

1) Client Scheduling

Similar to the insight of client selection algorithms mentioned in Section. III-C1 and illustrated in Fig. 6, randomly selecting clients is not the optimal solution for both statistical and system heterogeneity. For system heterogeneity, it is intuitive that making stragglers participate less in training can effectively reduce the time cost of each round. With the above intuition, Nishio et al. [118] proposed FedCS, which solves the client selection problem in a greedy fashion. Specifically, FedCS sets a certain deadline for each round and assumes that the server can obtain all clients' resource information, such as bandwidth and computation speed. With the given deadline, the client scheduler uses a heuristic algorithm to select as many clients which can complete the local training task in time as possible. Empirically, the FedCS algorithm usually selects the top k fast clients in each round. Mourad [119] proposed FedMCCS, an enhanced FL with multicriteria client selection. FedMCCS efficiently leverages stratified-based sampling to filter the available clients and maximize the number of clients participating in each FL round while considering their heterogeneity and limited communication and computation resources. However, both FedCS and FedMCCS may introduce bias into the global model due to the highly biased client selection of the heuristic algorithms.

Compared with the methods maximizing the number of clients greedily, FLANP, proposed by Reisizadeh et al. [120], better balances client participation while solving the straggler issue. The key idea of FLANP is to start the training procedure with faster nodes and gradually involve the slower nodes in the model training once the statistical accuracy of the data corresponding to the current participating nodes is reached. Although the time cost of the last few rounds of FLANP is relatively high, it allows slower clients to contribute to the model to achieve better accuracy. Chai et al. [121] proposed TiFL, a Tier-based Federated Learning System. To alleviate the impact of stragglers, TiFL divides clients into tiers based on their hardware resources, where homogeneous clients are divided into the same tier. Then in each round, TiFL selects clients from the same tier so that the stragglers do not slow down the training efficiency. Moreover, TiFL also takes the non-IID data into account. To be more specific, TiFL employs an adaptive tier selection approach to update the tiering on-the-fly based on the observed training performance and accuracy. Furthermore, reinforcement learning is also introduced into FL to solve system heterogeneity. Kim et al. [122] proposed AutoFL, a reinforcement learning algorithm that learns and determines which clients are selected each round. By carefully designing the status, action, and reward, AutoFL can select participant clients expected to maximize FL's energy efficiency while satisfying the accuracy requirement.

2) Workload Offloading

Workload balancing, widely used in distributed system and distributed learning, is the ability to distribute workload evenly (or based on clients' capabilities). In distributed learning, we can achieve workload balancing by simply reallocating data samples. However, this method can not be directly applied in FL since the private data can only be accessed by the owner. To offload workload under privacy guarantee, some works combine split learning (SL) [123], [124] with FL and allow clients to offload partial computation cost to the powerful server.

SL is a distributed machine learning approach that enables clients to collaborate to train a model without sharing raw data, just like FL. Besides, SL provides better model privacy than FL since the model splits between clients and servers. In addition, SL splits the entire model into multiple smaller subnetworks and trains them separately; hence SL is a better option for resource-constrain clients. Specifically, in a simple vanilla setting (Fig. 8(a)), a model is split into two portions: client-side model and server-side model. Each client trains the client-side model up to its last layer, known as the cut layer. Then the output of the cut layer, called smashed data, is sent to the server, which completes the rest of the forward and corresponding backward propagation. Finally, the gradients at the cut layer are sent back to the client to complete the rest of the backward propagation. This process is repeated by all clients sequentially until the model converges. Despite the advantages of SL, there is a primary limitation. The relay-based training in SL causes only one client to participate in training at a time. Such low parallelism of SL causes a significant increase in time overhead. Currently, some researches focus on designing an efficient system that unites the primary strength of FL (parallel processing among distributed clients) and SL (model splitting for less computation cost of clients) and eliminating their inherent drawbacks.

Ye et al. [125] proposed EdgeFed, which separates the process of updating the local model that is supposed to be completed independently by clients. Specifically, EdgeFed comprises clients, edge servers, and the central server. Each round includes multiple split training between clients and corresponding edge servers and a global aggregation between edge servers and the central server. First, the clients perform training on the client-side model and then send the smashed data to the edge server. After the remaining calculation, the edge server updates the whole model and returns the model weights of the client-side model to clients instead of the gradient of the cut layer. Finally, after several split training, the edge servers send the updated model weights to the central server for the weighted averaging to obtain the aggregated model. In short, EdgeFed introduces SL to offload workload from clients to the edge server at the cost of communication. In addition, the edge servers are introduced to play as a mediator to guarantee the parallelism of EdgeFed and training efficiency. With a similar design, Ullah et al. [126] proposed FedFly, a distributed FL+SL system with three layers architecture. Besides offloading workload and increasing parallelism, FedFly addresses the mobility challenges of clients in edge-based FL.

Compared with [125], [126], which uses weighted averaging to update the entire global model, SplitFed [127], the state-of-the-art FL+SL framework, utilizes the edge server to reduce the communication cost of uploading model weights.

To be more specific, the edge servers are responsible for offloading partial workload from clients and updating the server-side model with gradient descent every round, while the central server is in charge of only updating the client-side model with weighted averaging of model weights. Since clients only need to send the client-side model, which is a small portion of the model, the communication cost for uploading is further reduced. In addition, SplitFed introduces differential privacy and PixelDP to avoid privacy leakage while transmitting the smashed data and model weights.

However, the above works [125]–[127] bring another question: how much workload should be offloaded by each client. To answer this question, Ji et al. [128] proposed EAFL, which uses a threshold-based offloading strategy to reduce the computational burdens for stragglers in federated learning. Specifically, the strategy determines non-offloading and optimal partial offloading for clients with offloading decision indicator below and above a given threshold, respectively. FedAdapt, proposed by Wu et al. [129], adopts reinforcement learning-based optimization and clustering to adaptively identify which layers of the model should be offloaded for each client to a server to tackle the challenges of computational heterogeneity and changing network bandwidth.

C. SUMMARY AND LESSON LEARNED

In this section, we have discussed learning-based and protocol-based approaches for system heterogeneity. We summarize and compare the techniques with respect to their key ideas, disadvantages, and limitations, as shown in Table. 3.

Learning-based approaches aim to tackle straggler issues by modifying the learning algorithms to reduce the impact of stragglers or clients' workloads. Asynchronous training methods enable the global model can be updated without waiting for stragglers. However, the stale model updates from the stragglers may hurt the model performance and slow the convergence. Although several correction terms based on staleness [84], [85] and semi-asynchronous approaches [87], [88] are proposed, staleness and the trade-off of the degree of asynchronous are still open problems for FL. In addition, asynchronous methods suffer communication bottlenecks as all clients can communicate with the server asynchronously. Furthermore, the trusted execution environment [130] required by FedBuff [90] is only sometimes available in FL. Hence, aggregating individual model updates could still result in an undesirable level of privacy for FL. The limitation of heterogeneous epochs is similar to the privacy concern of asynchronous approaches. The privacy may leak since some works [93], [94] need to operate individual model updates. Model compression methods are the most efficient for reducing communication costs. However, the reduction of communication costs is at the expense of additional computational costs for compressing model weights. In addition, the relationship between compression rate and accuracy reduction remains to be seen. To improve training efficiency while not degrading accuracy, How to dynamically decide the optimal compression rate for each client needs to be clarified in the future. Partial model training breaks the assumption that the model architectures of all local models are the same and further achieves model heterogeneity. Nevertheless, the works based on unstructured dropout/pruning methods [107], [113], [115] may bring less efficiency improvement due to the limited support of ML frameworks or hardware. In addition, a trade-off exists between model performance and the strategy of deciding subnetworks. Indeed, assigning smaller subnetworks to clients implicitly reduces their contribution to the global model and can introduce bias into the global model. How partial model training approaches affect the model performance is an open problem left for future work.

Protocol-based approaches aim to solve the straggler issue by modifying the client selection schemes or trying to offload the stragglers' workload. The intuition of client scheduling is that limiting the time stragglers participate in training can improve training efficiency. However, such biased client selection may hurt the convergence and model performance [118]. Motivated by the observation, Huang et al. [131] shown that the fairness of client selection is essential for model performance in FL. Therefore, it is a challenge to design a client scheduling that improves training efficiency while guaranteeing the fairness of client selection. In addition, the server may not acquire all clients' statuses and suffer computation and communication overheads due to the complexity of scheduling many clients. Workload offloading utilizes split learning to offload partial computation overhead to the server. However, the reduction in computation overhead is at the expense of additional communication overhead (smashed data and gradient). In addition, the process that exchanges smashed data and corresponding gradient may leak data privacy.

V. FEDERATED LEARNING UNDER BOTH STATISTICAL AND SYSTEM HETEROGENEITIES

In Section. III and Section. IV, we have discussed the works on statistical and system heterogeneities, respectively. Actually, the two heterogeneities usually coexist in the FL environment, and it is urgent to design approaches tackling the two heterogeneities simultaneously. However, as the challenges move from single objective optimization to multiple objectives optimization, there may exist trade-offs between improving model performance (solving non-IID data) and increasing training efficiency (solving straggler issue). In this section, we first analyze the trade-offs and point out the potential exacerbation for heterogeneity brought by existing works. Then, we present the advance works considering both heterogeneities simultaneously.

A. LIMITATION OF EXISTING WORKS UNDER MULTIPLE HETEROGENEITIES

Although previous works perform well on a single heterogeneity, few of them have explored whether the proposed methods are robust enough for other types of heterogeneity.

Method		Key Ideas	Disadvantages and Limitations
Learning-based	Asynchronous Training	Designing an algorithm to enable the global model could be updated without waiting for stragglers.	• Staleness issue from the out-of-date updates of the stragglers hurts the model performance. • Asynchronous methods can easily create a network communication bottleneck. • Although various semi-asynchronous methods are proposed, the trade-off between training efficiency and accuracy is still unclear. • Asynchronous algorithms may not be compatible with secure aggregation and further leak privacy.
	Heterogeneous epochs	Adjusting the number of local epochs each client performs to re-balance their workload.	• The heterogeneous model progresses may hurt the convergence and model performance. • Most existing methods need to re-scale each model update; hence they may leak privacy to the honest-but-curious server.
	Model compression	Compressing the message transmitted between clients and the server to reduce communication costs. In addition, compression rates can be dynamically set to relieve the compression error.	• The compression error usually degrades model accuracy, and the trade-off between communication cost reduction and accuracy is still unclear. • The reduction in communication costs is at the expense of increased computational costs. • How to design compression methods fitting the FL scenario is still an open problem.
	Partial Model Training	Let clients perform training on heterogeneous models with different complexity to balance workload but still generate a single global model.	• Since ML frameworks, such as Pytorch, have limited support for sparse matrix computation, the works based on unstructured pruning/dropout methods may bring less improvement. • Finding the optimal subnetwork for each client is still an open problem. • The methods may lead to a biased model with poor accuracy, and the trade-off between cost reduction and accuracy needs to be clarified.
Protocol-based	Client scheduling	The training efficiency can be improved by making stragglers participate less in training or clustering clients based on their capabilities.	• The methods may introduce bias into the global model due to the biased client selection. • The assumption that all clients' statuses are known is not feasible in the real world. • Scheduling amount of clients may bring considerable overhead to the server.
	Workload Offloading	Utilizing split learning to offload stragglers' workload to the other powerful server or even clients and further achieve workload balancing.	• The reduction in computational costs is at the expense of increased communication costs. • How much workload each client should offload is still an open problem. • The communication pattern brings privacy issues and the trade-off between privacy and utility.

TABLE 3: Summary of techniques in *Taxonomy of System Heterogeneity*

Moreover, they may further make solving another heterogeneity more difficult. In this subsection, we present the implicit limitations of existing works and the interplay between proposed algorithms and heterogeneities.

1) Causes of Worsening System Heterogeneity

First, we review the proposed methods on statistical heterogeneity and discuss the reason they implicitly hurt system heterogeneity.

- *Increased Latency Disparity due to Additional Overheads*: In the case that the capabilities of each client remain unchanged, the gap between clients' response latency is proportional to the total overhead of a local training task. Unfortunately, the majority of previous works for statistical heterogeneity bring more overheads to clients compared with FedAvg. It makes clients which are already slow in response spend relatively more time completing training and aggravates the straggler issue from system heterogeneity. For example, the regularization methods mentioned in Section. III-B, including FedProx [7] and MOON [43], incurs $2 \sim 3$x overheads in respect of both memory and computation due to the requirements of regularization terms. In addition, the personalization methods, which apply existing DL techniques to help clients better transfer helpful knowledge from others, also burden the clients. For instance, despite some approximation approaches [61], [62] being proposed to help compute hessian terms, meta-learning algorithms are still computationally prohibitive. The methods that increase latency disparity include, but are not limited to, the examples mentioned above. How to effectively solve statistical heterogeneity without increasing overhead or even reducing it still challenges the research community.
- *Increased Time Cost due to Inappropriate Communication Protocol*: In Section. III-C1, we discussed algorithms that actively schedule clients' participation to counterbalance the bias introduced by non-

IID data. However, the clients that are helpful for the global model performance may be the stragglers which hurt training efficiency. More specifically, the previous works [77], [78] only pay attention to the clients that contribute to model performance and prioritize them. However, the factors they consider, such as local loss and local data distribution, are entirely different from that of the works for system heterogeneity [118], [119], including network bandwidth and computational capacities. Therefore, the client scheduling strategies for different heterogeneities may conflict and worsen each other. Besides client scheduling, the way clients communicate may also exacerbate system heterogeneity. For example, in Astraea [22], the clients are divided into groups, and then the clients in the same group perform local training sequentially. Although sequentially updating the global model does help tackle non-IID, it makes the straggler issue more severe and significantly endangers training efficiency. In addition, the peer-to-peer communication pattern used in FedCurv [47] brings expensive communication and time overheads. In each synchronization session, each client has to wait for all clients to complete transmission with itself. This is not feasible in an FL environment with heterogeneous capabilities and unstable network connections. For works tackling statistical heterogeneity, it is also important to consider communication protocol design and compatibility with other heterogeneities.

2) Causes of Worsening Statistical Heterogeneity

Next, we review the proposed methods on system heterogeneity and discuss the reason they implicitly hurt statistical heterogeneity.

- *Weight Divergence due to Reducing Workload*: With the key insight similar to FedProx [7], reducing workload for stragglers and naively incorporating partial information from them implicitly increase statistical heterogeneity. Specifically, reducing clients' workload means only partial local training tasks are completed, or partial model updates are transmitted. This limits the amount of available information contributing to the global model and induces the global model to bias towards the clients performing complete training, implying more significant weight divergence. Related issues have been explored in recent works mentioned in Section. IV-A2. As shown in the papers [7], [93], [94], tolerating heterogeneous local epochs lead to inconsistent step sizes of model updates and enhances the degree of weight divergence. In addition, although the trade-off between model compression rate and model performance has been noticed and well-studied in centralized training, it is still unclear in the FL environment due to the additional interaction between model compression and non-IID data. Currently, model compression robust to the feature of FL, such as non-IID and unstable client participation, is a popular research topic [95], [98], [99]. Moreover, the partial model training methods mentioned in Section. IV-A4 enable stragglers to perform training on subnetworks with less complexity. However, the methods usually have the challenge of accuracy-time trade-off. The greater the difference in model size between fast clients and stragglers, the greater the risk of model weight disparity. Although several works [103], [114], [115], [117] have empirically shown that their methods are pretty robust for different dropout/pruning rates, whether accuracy-time trade-off can be solved well in the real world environment remains to be clarified.
- *Decreased Fairness due to Biased Client Participation*: Similar to reducing the workload of stragglers, the difference in the frequency of participating in training also makes the global model biased towards the clients completing more local training tasks. Such a global model with bias may perform poorly on the stragglers, which participate less in training and violate the requirement of fairness that the global model should perform well on all clients. For asynchronous FL approaches, the number of times each client participates in training is proportional to his capabilities. Furthermore, the model weight uploaded by stragglers may be re-weighted or discarded due to staleness. These mechanisms have exacerbated the degree of deviation and the reduction of accuracy of the global model. In addition, the client scheduling algorithms for system heterogeneity also have similar issues. For example, the works [118], [119] with highly biased client selection methods ignore the importance of stragglers' local datasets and make them participate less in training. Similar approaches have been shown to reduce accuracy and slow down convergence significantly. Recently, Huang et al. [131] was aware of the disadvantages of biased client participation and empirically showed the importance of fairness in the client participation rate.

B. FEDERATED LEARNING METHODS FOR MULTIPLE HETEROGENEITIES

Reviewing the state-of-the-art in the field, we find that there are advanced works considering both heterogeneities simultaneously. In this subsection, we would like to provide an overall picture of the methods which tackle multiple heterogeneities. Noteworthy, the main goal of the works for multiple heterogeneities is usually to solve system heterogeneity. They empirically found implicit pitfalls mentioned in the last subsection and avoided them. To solve system heterogeneity while not increasing the degree of weight divergence, the works attempt to show the unbiasedness of the methods, correct the biased model updates, or model the relationship between heterogeneities. In the following content, we will discuss their key insights and how they tackle the interplay mentioned in the last subsection.

1) Asynchronous Training for Both Heterogeneities

As mentioned in Section. IV-A1, asynchronous training suffers from the staleness issue. Most current asynchronous FL methods re-weight the stale model updates to reduce their adverse impact. However, the re-weighting methods may significantly reduce the contribution of stragglers to the global model and make the global model suffer more from weight divergence. Compared with them, Li et al. [132] develop an adaptive approximation method called AD-SGD. Specifically, AD-SGD applies the Taylor expansion to approximate the "up-to-date" gradient of stragglers from its stale gradient. In addition, an adaptive hyper-parameter controlling mechanism is also integrated into AD-SGD to reduce the bias of Taylor series approximation. By approximating the optimal gradient from the stale model updates instead of reducing their aggregation coefficient, AD-SGD has the potential to be more robust to non-IID data. Besides, frequently communicating with specific clients would also exacerbate statistical heterogeneity. To handle the issue brought by biased client participation, some asynchronous methods propose compensation mechanisms to achieve unbiased, balancing training. For example, FedAT [88], a FL algorithm with asynchronous tiers, dynamically assigns aggregation coefficients to tiers based on the number of times they upload model updates. In this way, although the slower tiers (stragglers) less update the global model, they are assigned more significant coefficients while performing aggregation, and the potential bias towards the faster tiers is avoided. Similarly, ASO-Fed [86] applies a dynamic learning step size for each client based on the time cost of past iterations, where a larger time cost implies a larger learning rate. By doing so, the contribution of stragglers to the global model can be compensated.

2) Heterogeneous Epochs for Both Heterogeneities

As explained earlier in Section. IV-A2, heterogeneous epoch causes the heterogeneity in the model progress and exacerbates weight divergence. Among the works enabling heterogeneous local epochs, FedProx [7] introduces a proximal term to pull the local models closer to the global model and alleviate the gap between the step sizes of local models. Compared with FedProx, FedNova [94] normalizes the gradient when averaging, instead of restricting them, to achieve unbiased model aggregation. The authors of FedNova claimed that FedProx only mitigates the heterogeneity in the model progress, but FedNova can eliminate it. In addition, similar to FedAT [88], Ruan et al. [93] enlarge the aggregation coefficient for the clients which perform less local epochs to ensure an unbiased gradient after aggregation.

3) Model Compression for Both Heterogeneities

As mentioned in Section. IV-A3, being robust to non-IID data is one of the critical requirements for model compression methods in FL. Among the compression methods, Sattler et al. [95] conducted a series of experiments to evaluate the robustness of existing compression methods to non-IID data. The results showed that top-k sparsification suffers the least from non-IID data. Sattler et al. use this observation as an outset to construct a more efficient compression method for FL. Furthermore, FAB-top-k [100] emphasizes the fairness among clients in the number of gradient elements contributed to the sparse global model, which is helpful for non-IID data and compression since compression methods may aggravate the deviation and unfairness of local models. In addition, FedCOMGATE [99] tackles a problem similar to SCAFFOLD [41] in the context of model compression. To be more specific, both of them enable each client to use an estimate of the global update direction to correct its local updates. However, FedCOMGATE additionally considered the effect of compression in theoretical analysis. Besides the above works, Cur et al. [102] factor the compression error and non-IID data into the convergence analysis and explore the trade-off between model compression and model performance in the FL environment.

4) Partial Model Training for Both Heterogeneities

For the partial model training described in Section. IV-A4, because the methods still follow the standard FL process, which performs multiple local epochs under statistical heterogeneity, the subnetworks with different complexity suffer from weight divergence and digress to various scales. To relieve the weight divergence, some works design specific aggregation strategies. For example, To avoid disrupting the information of the subnetworks, Hermes [116] only averages the model weights that are intersected across the subnetworks of clients while keeping others unchanged instead of averaging the entire model of all clients. In addition, FedPrune [115] uses the insight from Lyapunov's Central Limit Theorem to perform aggregation. Specifically, FedPrune randomly draws aggregated model weights from the normal distribution with specific mean and variance and empirically shows the strategy's effectiveness. Besides aggregation strategies, based on the observation that dropout [108] scales representations based on dropout rate during the training phase to directly use the full model for inference, HeteroFL appends a scaler module right after the parametric layer. The evaluation results show that the global model composed of scaled subnetworks can achieve improved model performance.

5) Client Scheduling for Both Heterogeneities

As mentioned in Section. IV-B1, the highly biased client selection methods waste the potentially valuable data of stragglers and introduce bias into the global model. Guaranteeing fairness in participation is a crucial issue that is often overlooked. In FLANP [120], the stragglers are gradually involved in training instead of being abandoned. The improved fairness compared to other heuristic algorithms [118] is shown to help speed up the convergence in wall-clock time. Similarly, TiFL [121] dynamically adjusts the selection probability of each tier of clients to avoid heavily selecting certain tiers (e.g., tiers with faster clients). Moreover, Oort [133] formulates the trade-off between system and statistical heterogeneity and turns it into an optimization problem. Similarly, AutoFL [122] takes both heterogeneities into the reward function and tackles the optimization problem with a reinforcement learning approach.

C. SUMMARY AND LESSON LEARNED

In this section, we point out possible reasons of existing works for exacerbating another heterogeneity and review the methods considering both statistical and system heterogeneities simultaneously. The reason why the optimization for FL is challenging is its extremely complex learning environment and strict restrictions. When trying to solve statistical heterogeneity, we should pay attention to whether the proposed approaches cause a large amount of overhead or use inappropriate communication protocol, thus exacerbating the negative impact brought by system heterogeneity. Similarly, the works for system heterogeneity, especially tolerating partial workload and non-uniform client scheduling, need to take statistical heterogeneity into consideration to avoid exacerbating weight divergence or biasing the global model.

For the approaches considering both heterogeneities, compensating the model updates of specific clients is a widely used method. For instance, assigning larger learning rates [86] or aggregation coefficients [88], [93] to the clients which participate less in training or complete less workload can ensure unbiased updates after aggregation. In addition, to relieve weight divergence, some works correct the model updates by proximal terms [7] or estimate the 'better' update direction [94], [132]. Finally, fairness plays an important role in balancing heterogeneities. Although the definitions of fairness are different between different works, such as the number of gradients uploaded and the frequency of participating in training [120], [121], their theoretical or experimental results show that fairness is helpful for FL training.

VI. PROMISING FUTURE RESEARCH DIRECTIONS

Despite the research community's efforts, statistical and system heterogeneities are still open problems. Based on our survey of the existing heterogeneous FL literature, we would like to highlight a number of promising research directions, including heterogeneity analytics, fairness, FL architecture search, the Pareto Frontier of FL performance, and privacy issues.

A. PRIVACY CONCERNS

Although FL can provide a privacy guarantee for distributed systems by not sharing raw data during the training process, as shown in the works [134], [135], exchanging the model updates may still leak sensitive information to a third party. Even worse is that the relevant information about raw data (e.g., encoded data, GAN generators, and local data distributions) transmitted by data-based approaches has the opportunity to reveal more privacy. Although recent works adopted various privacy solutions, some challenges are still ahead. For example, differential privacy (DP) [136], [137] preserves training data and hidden information by inserting artificial noise, such as Gaussian noise, into the gradient or model weights. However, artificial noise can hurt the ability aggregating model updates accurately and brings the challenge of tackling the trade-off between privacy guarantee and model performance. On the other hand, encryption [138], [139] is considered lossless and does not degrade accuracy, but it can bring intensive computation and communication costs and further exacerbate the straggler issue. In addition, as mentioned in Section. IV-A1, asynchronous FL methods may be incompatible with secure aggregation and leak individual model updates to the server. Despite the effort of FedBuff [90], not all scenarios can support the Trusted Execution Environments [130], [140] required to implement FedBuff. In summary, privacy concerns have a strong link to the heterogeneity in FL. Therefore, it is an urge to design robust, low costs, practical solutions which simultaneously address heterogeneity and guarantee privacy.

B. HETEROGENEITY ANALYTIC

Most existing approaches have hyper-parameters that must be carefully adjusted to deal with varying degrees of heterogeneity. Furthermore, the model performance usually highly depends on hyper-parameter tuning. If we can accurately quantify heterogeneity over a federated environment before training starts, that would lead to better choices for hyper-parameters, model architecture design, and even FL algorithms. Recent works, such as FedProx [7], quantify statistical heterogeneity through local dissimilarity. However, these metrics can only be calculated during training instead of before starting. In sum, developing a simple analytic tool to quantify heterogeneity before training occurs quickly is essential for improving the convergence and performance of FL algorithms.

C. FAIR FEDERATED LEARNING

Fairness in FL is gaining attention in recent research. The general understanding of fairness in FL is that the global model performs well on all clients, which means the performance for all local datasets is similar and good enough. To avoid the global model biasing to specific clients, as mentioned in Section. V, current works ensure that each client at least participates in several training sessions or uploads several updates. However, unbiased FL training is challenging due to the diversity of clients, including the sizes of local datasets, activity patterns, hardware resources, and willingness to participate in training. While tackling statistical and system heterogeneities, existing work often needs to pay more attention to the importance of fairness. Although the model performance or training efficiency can be improved, the improvement is built on the sacrifice of partial clients. On the other hand, PFL is a promising direction for achieving fairness. Nevertheless, the study of PFL is still in its infancy. To make FL realistic and applicable, fair FL will become more and more critical.

D. FEDERATED LEARNING WITH NEURAL ARCHITECTURE SEARCH

In the presence of statistical heterogeneity, the predefined FL model architecture may not be the best choice. Neural Architecture Search (NAS) [141], [142] is a promising technique to help FL design better suitable model architecture for complex scenarios. Especially, Mendieta et al. [44] empirically shown that the Hessian eigenvalue and Hessian trace, which are significant predictors of performance and network generality in NAS [143], [144], help to determine the model architectures' robustness for non-IID data. Moreover, FedNAS [145] is proposed to enable scattered clients to search for better architecture collaboratively to achieve higher accuracy. On the other hand, NAS is the potential to automate the architecture design for the methods enabling model heterogeneity, such as PFL, and achieve better personalization.

E. MULTIPLE HETEROGENEITIES AND THE PARETO FRONTIER

As mentioned in Section. V, dealing with multiple heterogeneities is challenging. It is important to understand how these heterogeneities interact and systematically analyze the trade-off between them. According to the observation of this survey, model performance and training efficiency are in an intense relationship. Considering the effect of other heterogeneities while solving the main target is essential; otherwise, it may only find one of the Pareto optimal solutions. In other words, it can not bring improvement without detriment. In summary, improving the Pareto frontier, i.e., achieving better accuracy under the same costs or wall-clock time, is a popular but unsolved challenge in the research field.

VII. CONCLUSION

In this survey, we provide an overview of FL and present an in-depth and in-breadth investigation of FL with statistical and system heterogeneities. To the best of our knowledge, this survey is the first to discuss FL under multiple heterogeneities. Based on the key insights and techniques, we propose a unique taxonomy to categorize existing works for heterogeneous FL and highlight their features, limitations, and opportunities. Furthermore, we point out possible reasons of existing works for exacerbating another type of heterogeneity and systematically review the works for multiple heterogeneities. From the comprehensive review, several main lessons learned have been summarized and analyzed. Finally, we outline the research direction worth future effort. We believe that our taxonomy and the discussion of multiple heterogeneities will stimulate more attention in this emerging area and provide comprehensive insights to the research community.

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pages 1273–1282. PMLR, 2017.

[2] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. IEEE Transactions on Knowledge and Data Engineering, 2021.

[3] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. IEEE Communications Magazine, 58(6):46–51, 2020.

[4] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 23(3):1622–1658, 2021.

[5] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133, 2020.

[6] Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. Robust and communication-efficient collaborative learning. Advances in Neural Information Processing Systems, 32, 2019.

[7] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2:429–450, 2020.

[8] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2):1–19, 2019.

[9] Sawsan AbdulRahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. IEEE Internet of Things Journal, 8(7):5476–5497, 2020.

[10] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. IEEE Access, 8:140699–140725, 2020.

[11] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3):50–60, 2020.

[12] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. Knowledge-Based Systems, 216:106775, 2021.

[13] Yi Liu, Xingliang Yuan, Zehui Xiong, Jiawen Kang, Xiaofei Wang, and Dusit Niyato. Federated learning for 6g communications: Challenges, methods, and future directions. China Communications, 17(9):105–118, 2020.

[14] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 22(3):2031–2063, 2020.

[15] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges. IEEE Internet of Things Journal, 8(16):12806–12825, 2021.

[16] Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. IEEE Communications Surveys & Tutorials, 2021.

[17] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. Future Generation Computer Systems, 115:619–640, 2021.

[18] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. IEEE Transactions on Neural Networks and Learning Systems, 2022.

[19] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189, 2019.

[20] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018.

[21] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. International journal of computer vision, 40(2):99, 2000.

[22] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Transactions on Parallel and Distributed Systems, 32(1):59–71, 2020.

[23] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In ICC 2020-2020 IEEE International Conference on Communications (ICC), pages 1–7. IEEE, 2020.

[24] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357, 2002.

[25] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pages 1322–1328. IEEE, 2008.

[26] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2):539–550, 2008.

[27] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621, 2017.

[28] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1542–1547. IEEE, 2018.

[29] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019.

[30] Weituo Hao, Mostafa El-Khamy, Jungwon Lee, Jianyi Zhang, Kevin J Liang, Changyou Chen, and Lawrence Carin Duke. Towards fair federated learning with zero-shot data augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3310–3319, 2021.

[31] MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. arXiv preprint arXiv:2006.05148, 2020.

[32] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. arXiv preprint arXiv:2107.00233, 2021.

[33] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. Communications of the ACM, 63(11):139–144, 2020.

[35] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv preprint arXiv:1811.11479, 2018.

[36] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[37] Mu Yan, Bolun Chen, Gang Feng, and Shuang Qin. Federated cooperation and augmentation for power allocation in decentralized wireless networks. IEEE Access, 8:48088–48100, 2020.

[38] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In International conference on machine learning, pages 214–223. PMLR, 2017.

[39] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, and Albert Y Zomaya. Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. IEEE Internet of Things Journal, 2021.

[40] Longling Zhang, Bochen Shen, Ahmed Barnawi, Shan Xi, Neeraj Kumar, and Yi Wu. Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. Information Systems Frontiers, 23(6):1403–1415, 2021.

[41] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning, pages 5132–5143. PMLR, 2020.

[42] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. arXiv preprint arXiv:2111.04263, 2021.

[43] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10713–10722, 2021.

[44] Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8397–8406, 2022.

[45] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

[46] Kavya Kopparapu and Eric Lin. Fedfmc: Sequential efficient federated learning on non-iid data. arXiv preprint arXiv:2006.10937, 2020.

[47] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. arXiv preprint arXiv:1910.07796, 2019.

[48] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. In 2020 IEEE International Conference on Image Processing (ICIP), pages 1736–1740. IEEE, 2020.

[49] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. Journal of Big data, 3(1):1–40, 2016.

[50] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. IEEE Intelligent Systems, 35(4):83–93, 2020.

[51] Hongwei Yang, Hui He, Weizhe Zhang, and Xiaochun Cao. Fedsteg: A federated transfer learning framework for secure image steganalysis. IEEE Transactions on Network Science and Engineering, 8(2):1084–1094, 2020.

[52] Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. arXiv preprint arXiv:1909.12535, 2019.

[53] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. arXiv preprint arXiv:1912.00818, 2019.

[54] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In International Conference on Machine Learning, pages 2089–2099. PMLR, 2021.

[55] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. arXiv preprint arXiv:2001.01523, 2020.

[56] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence, 44(9):5149–5169, 2021.

[57] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pages 1126–1135. PMLR, 2017.

[58] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488, 2019.

[59] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999, 2(3):4, 2018.

[60] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876, 2018.

[61] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835, 2017.

[62] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. Advances in Neural Information Processing Systems, 33:3557–3568, 2020.

[63] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. Advances in Neural Information Processing Systems, 33:21394–21405, 2020.

[64] Younghyun Park, Dong-Jun Han, Do-Yeon Kim, Jun Seo, and Jaekyun Moon. Few-round learning for federated learning. Advances in Neural Information Processing Systems, 34:28612–28622, 2021.

[65] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. Advances in neural information processing systems, 30, 2017.

[66] Luca Corinzia, Ami Beuret, and Joachim M Buhmann. Variational federated multi-task learning. arXiv preprint arXiv:1906.06268, 2019.

[67] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In International Conference on Machine Learning, pages 6357–6368. PMLR, 2021.

[68] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In AAAI, pages 7865–7873, 2021.

[69] Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang. Personalized federated learning with graph. arXiv preprint arXiv:2203.00829, 2022.

[70] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[71] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. Advances in Neural Information Processing Systems, 34:15434–15447, 2021.

[72] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581, 2019.

[73] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10143–10153, 2022.

[74] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In International Conference on Machine Learning, pages 12878–12889. PMLR, 2021.

[75] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems, 33:2351–2363, 2020.

[76] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. Advances in Neural Information Processing Systems, 33:14068–14080, 2020.

[77] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv preprint arXiv:2010.01243, 2020.

[78] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pages 1698–1707. IEEE, 2020.

[79] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE transactions on neural networks and learning systems, 32(8):3710–3722, 2020.

[80] Christopher Briggs, Zhong Fan, and Peter 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), pages 1–9. IEEE, 2020.

[81] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. arXiv preprint arXiv:2005.01026, 2020.

[82] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

[83] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. Advances in Neural Information Processing Systems, 33:19586–19597, 2020.

[84] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. arXiv preprint arXiv:1903.03934, 2019.

[85] Xiaofeng Lu, Yuying Liao, Pietro Lio, and Pan Hui. Privacy-preserving asynchronous federated learning mechanism for edge network computing. IEEE Access, 8:48970–48981, 2020.

[86] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In 2020 IEEE International Conference on Big Data (Big Data), pages 15–24. IEEE, 2020.

[87] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. IEEE Transactions on Computers, 70(5):655–668, 2020.

[88] Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. ArXivorg, 2020.

[89] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1175–1191, 2017.

[90] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In International Conference on Artificial Intelligence and Statistics, pages 3581–3607. PMLR, 2022.

[91] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 308–318, 2016.

[92] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. Papaya: Practical, private, and scalable federated learning. Proceedings of Machine Learning and Systems, 4:814–832, 2022.

[93] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. Towards flexible device participation in federated learning. In International Conference on Artificial Intelligence and Statistics, pages 3403–3411. PMLR, 2021.

[94] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. Advances in neural information processing systems, 33:7611–7623, 2020.

[95] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems, 31(9):3400–3413, 2019.

[96] Jed Mills, Jia Hu, and Geyong Min. Communication-efficient federated learning for wireless edge intelligence in iot. IEEE Internet of Things Journal, 7(7):5986–5994, 2019.

[97] Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. Federated learning with quantization constraints. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 8851–8855. IEEE, 2020.

[98] Jinjin Xu, Wenli Du, Yaochu Jin, Wangli He, and Ran Cheng. Ternary compression for communication-efficient federated learning. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[99] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In International Conference on Artificial Intelligence and Statistics, pages 2350–2358. PMLR, 2021.

[100] Pengchao Han, Shiqiang Wang, and Kin K Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pages 300–310. IEEE, 2020.

[101] Liang Li, Dian Shi, Ronghui Hou, Hui Li, Miao Pan, and Zhu Han. To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pages 1–10. IEEE, 2021.

[102] Laizhong Cui, Xiaoxin Su, Yipeng Zhou, and Jiangchuan Liu. Optimal rate adaption in federated learning with compressed communications. In IEEE INFOCOM 2022-IEEE Conference on Computer Communications, pages 1459–1468. IEEE, 2022.

[103] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. arXiv preprint arXiv:2010.01264, 2020.

[104] Hakim Sidahmed, Zheng Xu, Ankush Garg, Yuan Cao, and Mingqing Chen. Efficient and private federated learning with partially trainable networks. arXiv preprint arXiv:2110.03450, 2021.

[105] Jae Hun Ro, Theresa Breiner, Lara McConnaughey, Mingqing Chen, Ananda Theertha Suresh, Shankar Kumar, and Rajiv Mathews. Scaling language model size in cross-device federated learning. arXiv preprint arXiv:2204.09715, 2022.

[106] Tien-Ju Yang, Dhruv Guliani, Françoise Beaufays, and Giovanni Motta. Partial variable training for efficient on-device federated learning. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4348–4352. IEEE, 2022.

[107] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. Communication-efficient federated learning with adaptive parameter freezing. In 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pages 1–11. IEEE, 2021.

[108] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

[109] Russell Reed. Pruning algorithms-a survey. IEEE transactions on Neural Networks, 4(5):740–747, 1993.

[110] Xue Ying. An overview of overfitting and its solutions. In Journal of physics: Conference series, volume 1168, page 022022. IOP Publishing, 2019.

[111] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635, 2018.

[112] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. arXiv preprint arXiv:1812.07210, 2018.

[113] Zirui Xu, Zhao Yang, Jinjun Xiong, Janlei Yang, and Xiang Chen. Elfish: Resource-aware federated learning on heterogeneous edge devices. Ratio, 2(r1):r2, 2019.

[114] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems, 34:12876–12889, 2021.

[115] Muhammad Tahir Munir, Muhammad Mustansar Saeed, Mahad Ali, Zafar Ayyub Qazi, and Ihsan Ayyub Qazi. Fedprune: Towards inclusive federated learning. arXiv preprint arXiv:2110.14205, 2021.

[116] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 420–437, 2021.

[117] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. IEEE Transactions on Neural Networks and Learning Systems, 2022.

[118] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In ICC 2019-2019 IEEE international conference on communications (ICC), pages 1–7. IEEE, 2019.

[119] Sawsan AbdulRahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. Fedmccs: Multicriteria client selection model for optimal iot federated learning. IEEE Internet of Things Journal, 8(6):4723–4735, 2020.

[120] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. arXiv preprint arXiv:2012.14453, 2020.

[121] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, pages 125–136, 2020.

[122] Young Geun Kim and Carole-Jean Wu. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, pages 183–198, 2021.

[123] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564, 2018.

[124] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications, 116:1–8, 2018.

[125] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. Edgefed: Optimized federated learning based on edge computing. IEEE Access, 8:209191–209198, 2020.

[126] Rehmat Ullah, Di Wu, Paul Harvey, Peter Kilpatrick, Ivor Spence, and Blesson Varghese. Fedfly: Towards migration in edge-based distributed federated learning. IEEE Communications Magazine, 2022.

[127] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 8485–8493, 2022.

[128] Zhongming Ji, Li Chen, Nan Zhao, Yunfei Chen, Guo Wei, and F Richard Yu. Computation offloading for edge-assisted federated learning. IEEE Transactions on Vehicular Technology, 70(9):9330–9344, 2021.

[129] Di Wu, Rehmat Ullah, Paul Harvey, Peter Kilpatrick, Ivor Spence, and Blesson Varghese. Fedadapt: Adaptive offloading for iot devices in federated learning. IEEE Internet of Things Journal, 2022.

[130] Ryan Karl, Jonathan Takeshita, Nirajan Koirla, and Taeho Jung. Cryptonite: a framework for flexible time-series secure aggregation with online fault tolerance. Cryptology ePrint Archive, 2020.

[131] Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. IEEE Transactions on Parallel and Distributed Systems, 32(7):1552–1564, 2020.

[132] Xingyu Li, Zhe Qu, Bo Tang, and Zhuo Lu. Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients. arXiv preprint arXiv:2102.06329, 2021.

[133] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21), pages 19–35, 2021.

[134] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems, 33:16937–16947, 2020.

[135] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. IEEE transactions on neural networks and learning systems, 2022.

[136] Cynthia Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer, 2008.

[137] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security, 15:3454–3469, 2020.

[138] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint arXiv:1711.10677, 2017.

[139] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In 2020 USENIX annual technical conference (USENIX ATC 20), pages 493–506, 2020.

[140] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: privacy-preserving federated learning with trusted execution environments. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, pages 94–108, 2021.

[141] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.

[142] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. The Journal of Machine Learning Research, 20(1):1997–2017, 2019.

[143] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. arXiv preprint arXiv:1909.09656, 2019.

[144] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In International conference on machine learning, pages 1554–1565. PMLR, 2020.

[145] Chaoyang He, Erum Mushtaq, Jie Ding, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. 2021.