Towards Efficient Synchronous Federated Training: A Survey on System Optimization Strategies

Zhifeng Jiang, *Student Member, IEEE,* Wei Wang, *Member, IEEE,* Bo Li, *Fellow, IEEE,* Qiang Yang, *Fellow, ACM*

Abstract—The increasing demand for privacy-preserving collaborative learning has given rise to a new computing paradigm called federated learning (FL), in which clients collaboratively train a machine learning (ML) model without revealing their private training data. Given an acceptable level of privacy guarantee, the goal of FL is to minimize the *time-to-accuracy* of model training. Compared with distributed ML in data centers, there are four distinct challenges to achieving short time-to-accuracy in FL training, namely the lack of information for optimization, the tradeoff between statistical and system utility, client heterogeneity, and large configuration space. In this paper, we survey recent works in addressing these challenges and present them following a typical training workflow through three phases: client selection, configuration, and reporting. We also review system works including measurement studies and benchmarking tools that aim to support FL developers.

Index Terms—Federated Learning, Synchronous Training, Survey.

---

1 INTRODUCTION

BUILDING high-quality machine learning (ML) models demands a massive amount of training data. Yet, the communication cost and privacy concerns impinge on the process of collecting large volumes of data from diverse sources. It is not until recently that governments started to regulate the commercial use of data with privacy-preserving legislation (e.g., GDPR [1], HIPAA [2], and CCPA [3]). Compliance violations can be costly, with hefty fines up to hundreds of millions of dollars a year [4], [5]. As such, the desire for multiple entities (e.g., mobile devices or large organizations) to collaboratively train a shared model efficiently and privately gives birth to a new ML paradigm called federated learning (FL) [6]. FL promises not to expose the clients' raw data, and has been widely adopted in many industries with applications ranging from mobile devices [7], [8], [9], [10], [11], [12] to financial management [13], [14] and medical care [15], [16].

Apart from providing strong privacy guarantees, the key to the success of a federated training system lies in its efficiency. A typical efficiency metric is the *time-to-accuracy*, which is the wall clock time taken to train a model until it reaches the target accuracy. Despite the rich body of work that explored various optimization strategies, there is still plenty of room for further improvement due to the following distinct challenges posed to FL (§2): (1) the *lack of information for optimization*: the information needed for optimally configuring the system is usually outdated or simply unavailable due to privacy constraints and scaling issues;

(2) the *tradeoff between statistical and system utility*: statistical utility (the number of iterations taken to reach a plausible target accuracy) and system utility (the duration of an iteration), the two determining factor for time-to-accuracy, are usually at odds in FL; (3) *client heterogeneity*: clients cannot be treated uniformly due to the intrinsic differences in terms of resources, data, and states; and (4) a *large configuration space*: the operational dimensions for system developers are too many to explore within an acceptable time. Given these challenges, it is worth summarizing existing research efforts to give researchers a holistic view of the lessons learned and to solicit further explorations.

To position existing research attempts in optimizing the time-to-accuracy performance in FL, we propose a layered approach that categorizes them by the training phases in which they take effect: selection, configuration, and reporting (§3). For the *selection phase* where the server chooses clients for participation, there are mainly two lines of optimization efforts: (1) prioritizing clients either with high statistical utility or system utility [17], [18], [19], and (2) explicitly considering both utilities and developing a more informed solution in response to client dynamics in practice [20], [21], [22].

As for the *configuration phase* where the server sends the global model to the selected clients with auxiliary configuration information, and clients perform local training, we sort out four lines of work: (1) the first two lines advocate mitigating the communication cost by reducing the model size [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36] and decreasing the synchronization frequency [37], [38], [39], [40], [41]; (2) the last two lines minimize the computational overhead by accelerating the training speed in each round [42], [43], [44], [45], [46], [47] and reducing the number of training rounds [45], [48], [49],

---

• *Z. Jiang, W. Wang, B. Li and Q. Yang are with the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Q. Yang is also affiliated with WeBank.*
E-mail: {zjiangaj, weiwa, bli, qyang}@cse.ust.hk

[50], [51], [52], [53], [54], [55].

In terms of the *reporting phase*, we focus on the aggregation and outline two related optimizations: (1) reducing the aggregation latency by adopting hierarchical methods [56], [57], [58] and developing lightweight privacy-preserving methods [59], [60], [61], and (2) improving the long-term convergence rate with adaptive optimizers on the server [62], [63], [64]. For each of the attempted optimizations, our discussion includes necessary details for readers to understand the motivation, mechanisms, and major results. In addition, works such as measurement studies [65] and benchmarking tools [66], [67], [68], [69], [70], [71], [72] are indispensable in system research. We also survey the status quo as a tutorial on FL practice (§4).

Our work focuses on the system-level efforts made in improving the time-to-accuracy performance for synchronous federated training. We also share some implications derived from the literature and our survey process. It thus differs from existing surveys which mainly focus on ML algorithms and privacy-preserving algorithms. We expect this work to be an initial attempt to bridge the gap of system-oriented surveys in FL literature, as well as soliciting more contributions to related research.

2 BACKGROUND, PROBLEM AND CHALLENGES

In this section, we give a detailed introduction to the system optimization problem in federated training. We start with a quick primer on the execution workflow of federated training (§2.1), followed by the problem statement and the scope of this survey (§2.2). We next outline two challenges that make the problem difficult: optimality and practicality (§2.3), which also serves as a summary of criteria for evaluating existing solutions presented thereafter.

2.1 Federated Training

Federated learning (FL) [6] has recently emerged as a new paradigm of collaborative machine learning (ML) that allows multiple distributed clients (e.g., mobile devices or business organizations) to collaboratively train or evaluate a model with decentralized data. Compared to traditional distributed learning in datacenter environments, FL mainly differs in orchestration, resource constraints, data distribution, and participation scale [73]. At its core, FL keeps private data on-premises, while using a central server to maintain a global model and iteratively refine it by aggregating each client's local updates. This design reduces not only the communication cost but also the privacy risk in gathering clients' raw data. Owing to its privacy guarantees, FL has found wide applications in various domains. On mobile devices, Google runs FL to improve the user experience for Google Keyboard [7], [8], [9], [10] and Assistant [11], while Apple deploys FL to evaluate and tune speech recognition models [12]; in fintech, both IBM [13] and WeBank [14] utilize FL to detect financial misconducts; in healthcare, NVIDIA applies FL to create medical imaging AI [15] and predict patients' needs for oxygen [16].

While both model training and evaluation play important roles in the development of an FL model, they have different criteria in system design. In this survey, we limit
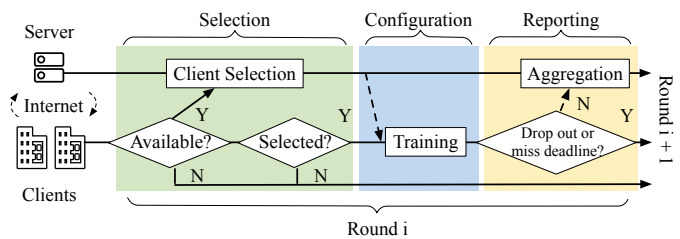

Fig. 1: Standard synchronous federated training protocol [65], [68], [74].

the scope to the *training* process, which is the most time-consuming and resource-intensive stage throughout the development of an FL model. Due to its predominance in practice, we focus on the support for the *synchronous* mode, wherein an ML model is trained across a pool of candidate clients in rounds, and in each round, the server needs to wait until a predefined deadline or receiving a sufficient number of clients' updates prior to deriving an aggregated update. In more detail, each round consists of the following three phases (Fig. 1).

- *Client Selection*. At the beginning of each round, the server waits for a sufficient number of clients with eligible status (i.e., currently charging and connected to an unmetered network) to check in. The server then selects a subset of them based on certain strategies (e.g., randomly or selectively) for participation, and notifies the others to reconnect later.
- *Configuration*. The server next sends the global model status and configuration profiles (e.g., the number of local epochs or the reporting deadline) to each of the selected clients. Based on the instructed configuration, the clients perform local model training independently with their private data.
- *Reporting*. The server then waits for the participating clients to report local updates until reaching the predefined deadline. The current round is aborted if no enough clients report in time. Otherwise, the server aggregates the received local updates, uses the aggregate to update the global model status, and concludes the round.

2.2 System Optimization: The Problem

The primary goal of system optimization in federated training is to minimize the end-to-end resource usage of performing a task. The most common metric is the wall clock time, which is typically measured from the very beginning to a certain desirable checkpoint (e.g., convergence or reaching target accuracy). When a metered network is in use (e.g., when clients are on-demand virtual machines in a public cloud), the overall monetary cost becomes another relevant metric that deserves special attention. When uncharged devices are involved, the power consumption should also be considered. Because the cost and energy consumption generally grows linearly as time flies, in this survey, we are particularly interested in reducing the *time-to-accuracy*, i.e., the wall clock time for achieving a preset accuracy target.

(a) Time to accuracy performance. (b) Round to accuracy performance. (c) Average round time w/ standard deviation. (d) Data distributions of clients in NonIID. (e) Computational speed of clients in Straggler.
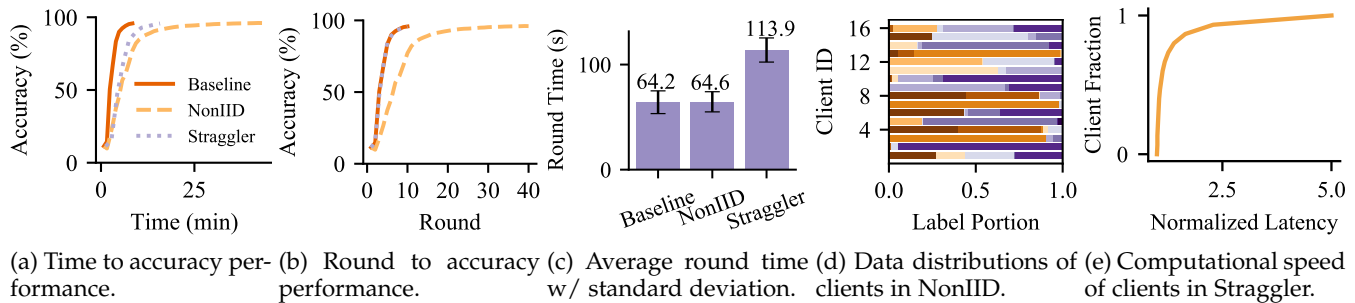
Fig. 2: A case study for illustrating system utility and statistical utility with FL emulation.

2.2.1 Statistical Utility and Sytem Utility: A Case Study

Intuitively, the time-to-accuracy performance of federated training is determined by two factors: the number of rounds taken to reach the accuracy and the average duration of training rounds. As in [21], we regard the former as *statistical utility* and the latter as *system utility* throughout this survey.

To illustrate, we conduct an FL case study where 16 clients collaboratively train the LeNet5 [75] model to classify the images from the MNIST [76] dataset. To emulate the cross-device environment, each client is run atop an AWC EC2 `c5.xlarge` instance (4 vCPUs and 8 GB memory) and the network bandwidth is throttled to 21 Mbps to dictate the average mobile connection speed as of 2021 [77]. We run FedAvg [6] for model aggregation without loss of generality, atop which we enforce clients' privacy by implementing the SecAgg protocol [78] (also see §3.2.3) which makes the server blind to each client's local updates. Essentially, we design three schemes to compare:

- *Baseline*: all clients' data is independent and identically distributed (IID). They also share the same computational speed.
- *NonIID*: the same as *Baseline* except that clients' data is non-IID as depicted by Fig. 2d. This is achieved by latent Dirichlet allocation (described in §4.2.1) with the concentration vectors being all 0.1's.
- *Straggler*: the same as *Baseline* except that clients' computation speeds follow the Zipf's distribution (with $\alpha = 1.2$, i.e., moderately skewed). As shown in Fig. 2e, the straggler is $5\times$ slower than the fastest who shares the same speed as clients in *Baseline*.

We first study the impact of statistical utility by comparing *Baseline* and *NonIID*. As indicated in Fig. 2c, they proceed a round at the same speed. However, as the learning curves in Fig. 2a dictate, *Baseline* reaches the target accuracy (96.1%) with only 8.4 min, while *NonIID* takes 43.5 min, which is $5\times$ slower than *Baseline* does. Referring to the round-to-accuracy performance as in Fig. 2b, one can know that the reason for this contrast lies in the difference in the numbers of rounds taken to convergence: 10 for *Baseline* while 40 for *NonIID*. Hence, despite having the same system utility, *Baseline* achieves much better time-to-accuracy performance for having greater statistical utility due to more evenly distributed data across clients. Of course, data distribution is uncontrollable in FL practice and developers can barely end up with IID cases as in *Baseline*. This adds up to the challenges of system-level optimizations in FL training, as later discussed in §2.3.1.

Next, we provide a sense of how system utility affects the end-to-end performance by comparing *Baseline* with *Straggler*. As shown in Fig. 2b, both cases converge with the same amount of rounds: 10. However, it takes *Straggler* slightly longer in time (15.8 min, i.e., $1.9\times$) to reach the target accuracy, as demonstrated in Fig. 2a. By observing Fig. 2b and Fig. 2c, we know that the source of the difference does not stem from the number of rounds used but the making span of each round. As all clients have to proceed at the same speed as the slowest clients in synchronous training, *Straggler* features lower system utility than *Baseline* does due to the presence of slow clients, rendering worse time-to-accuracy. Similarly, the disparity of clients' capabilities is inevitable in FL practice, which also complicates the design of system-level optimizations as mentioned in §2.3.1.

2.2.2 What is beyond the Scope

We emphasize that the solutions discussed in this survey operate at the *system-level*. Therefore, the following directions of work are excluded from discussion, though they could effectively improve the statistical and/or system efficiency in synchronous federated training:

- Hardware updates, e.g., adapting programmable switches to enjoy the communication efficiency brought by in-network aggregation [79], [80].
- Security mechanisms, e.g., employing robust aggregation methods to protect the statistical utility from being impaired by model poisoning attacks [81], [82].
- Paradigm innovations, e.g., (1) letting clients fit different sets of model parameters by personalization [83], [84], [85], [86], [87] or models of heterogeneous architectures through knowledge distillation [88], [89], [90] to tackle data heterogeneity (a concept mentioned in §2.3.1), (2) allowing clients to exchange data representations for realizing prototype learning [91] to reduce communication burdens, or (3) permitting clients to send encoded versions of local datasets to the server to reduce the computational complexity [92].
- Optimizations on upstream parts of the FL pipeline, e.g., searching for neural architectures that yield better predictive accuracy [93], [94].

2.3 What Makes It Hard: The Challenges

Despite the clear objective, it is non-trivial to work out a feasible solution due to the following two challenges.

2.3.1 On the Optimality of a Solution

First, the information needed in decision-making may be *outdated or even unavailable*. For example, to estimate a client's system utility, it is common to refer to its most recent response latency [21]. However, due to the dynamics over time, such information may not accurately reflect the client's current status. There also exists a cold-start issue, where we are unaware of a client's system capabilities until its first participation. As for estimating a client's statistical utility, the amount of available information is further limited by privacy concerns. According to the recent FL literature, exploratory attacks such as property inference [95], membership inference [96], [97], and data reconstruction [98], [99] can be made possible with model updates. As such, even exposing model updates can discourage clients from participation, let alone inquiring about their data distributions or even raw data [100], [101], [102]. Note that the uncertainty in clients' statistical utility and system utility can be accumulated over time.

Even given a holistic view of the environment, the problem remains hard due to the *coupled nature* of statistical utility and system utility. Intuitively, improving system utility is equivalent to minimizing the average resource consumption (e.g., time or bandwidth) per task unit. On the other hand, reducing the resources invested in a task unit inevitably downgrades the quality of the outcome (e.g., statistical utility) as long as no resource is redundant. To exemplify, by constantly picking the fastest clients in client selection, the average duration of each round indeed decreases, whereas the number of rounds taken to target accuracy may be increased as well when other clients' data are under-represented in the global model. Another example can be found in model compression. To improve communication efficiency, a client can send only an important subset of model updates by sparsification [32], [33], [34], [35], [36], or a low-bit representation of them by quantization [23], [24], [25], [26], [27]. Although the per-round communication duration can be significantly reduced by adopting a higher compression ratio, the convergence has to take more rounds to occur due to the loss of arithmetic precision.

The problem is further complicated by *client heterogeneity*. Federated training involves tens to potentially millions of clients, each of which intrinsically differs from one another in the following three aspects:

- *Resource Heterogeneity*: due to the variability in hardware specifications and system-level constraints, clients in federated training typically possess different capabilities in computation (CPU/GPU/NPU, memory, and storage), communication (connectivity and bandwidth) and power (battery level and lifespan) [103]. These types of heterogeneity complicate the optimization of the overall system utility. For example, merely improving the communication speed does not necessarily lead to shorter end-to-end latency, especially when the straggler is bottlenecked by the computation [68].
- *Data Heterogeneity*: as the training datasets of clients are typically generated based on their local activities and contexts, they are not IID. More specifically,

clients' datasets mainly differ in two aspects[1]: (1) sample quantity (i.e., the number of data samples), and (2) label partition (i.e., the distribution of data labels) [104]. As a result, not all of them are representative of the population distribution. In case we do not include all the clients in the federation, optimization for statistical utility has to additionally account for such heterogeneity.

- *State Heterogeneity*: as observed from real-world traces [65], [68], the available slots of mobile device clients vary significantly in temporal distribution due to different user behaviors (e.g., screen locking or battery charging). Therefore, in each round, there can be different sets of candidate clients to choose from, as well as different client drop-out outcomes. On top of the non-IID distribution of clients' data, this type of heterogeneity further complicates statistical utility optimization. Nevertheless, in the cross-silo settings, it may be less of a concern due to the stable and dedicated nature of clients' computing power [70], [73].

Last but not least, it is *infeasible to search through the entire configuration space* for the global optimum. On the one hand, the space is prohibitively large, as a federated training task typically spans 10^1–10^6 users and 10^2–10^4 rounds [73], wherein each phase of a round (§2.1) has multiple configurable hyperparameters and alternative policies (e.g., client selection choices in the selection phase, or the number of local steps in configuration phase). On the other hand, most of the online decisions are made on the critical path of the task, meaning that the time spent on working out a solution also counts towards the end-to-end runtime performance, the very objective of the optimization. As a result, it is desirable to be guided by efficient and effective heuristic algorithms, especially balancing the exploration and exploitation efforts made in the configuration space.

2.3.2 On the Practicality of a Solution

Apart from navigating the performance-accuracy-privacy trade-off, the design process of a practical optimization solution should mitigate the accompanying side-effects on other aspects such as the loss of *robustness* to attacks and failures [73]. For example, to evaluate the statistical utility of a client, the server may require it to report the loss values generated in local training [21]. However, a malicious or free-rider client may intentionally respond with arbitrary values in the hope of messing with the orchestration or reaping the benefits of the federation without making solid contributions. As such backdoors are introduced by the optimization solution, the developers should take charge of eliminating the undesirable exploitations of these security loopholes. Other possible concerns that may arise as a result of a system optimization solution include but are not limited to *fairness* (e.g., whether participant bias is introduced in the solution), *generality* (e.g., whether the solution applies to diverse tasks), and *ease of deployment* (e.g., whether the solution can be implemented with moderate engineering efforts). In other words, a mature system optimization so-

---

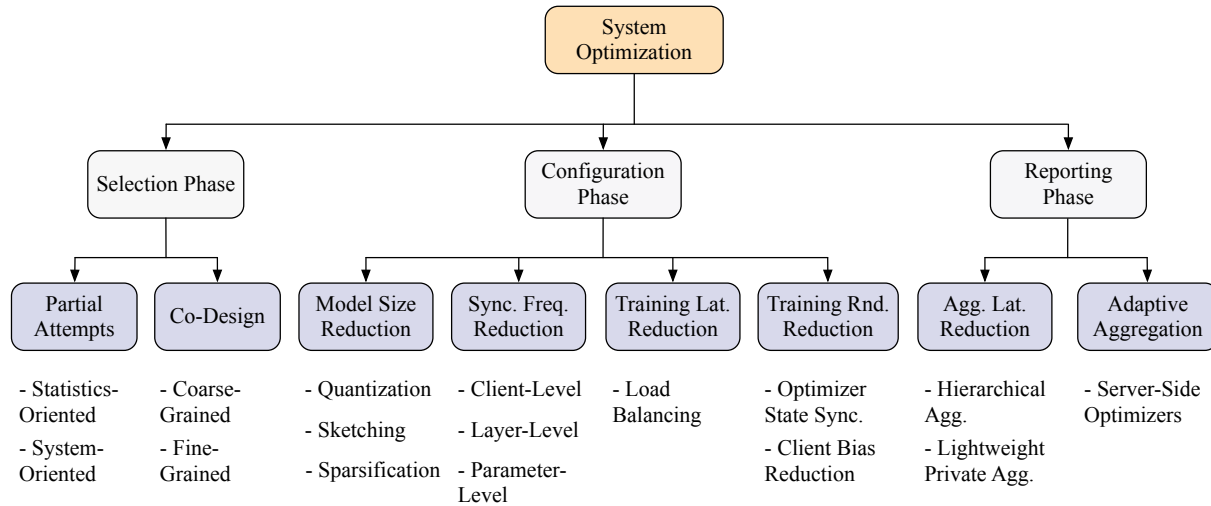1. See a more complete categorization of non-IID scenarios in §4.2.1.

Fig. 3: Taxonomy of the approaches discussed in §3.

lution should not only improve the time-to-accuracy performance by enhancing statistical and system utility but also minimize the adverse impacts on other aspects that federated training also values in practice.

3 RECENT OPTIMIZATION APPROACHES

In the past few years, considerable research efforts have been put into tackling the above challenges for fully unleashing the performance potential of FL training. In this section, we organize them by the training phases, i.e. selection (§3.1), configuration (§3.2), and reporting (§3.3), as visualized in Fig. 3.

3.1 Optimizing the Selection Phase

Due to the (potentially) large population size and the heterogeneity across clients, the effectiveness of the used participant selection algorithm plays a critical role in the time-to-accuracy performance in federated training. However, the state-of-the-practice system still relies on randomly picking participants [74], which inevitably leads to waste of resources and suboptimal convergence speed. In response, there is an array of work to guide the selection, which can be roughly categorized by the target utilities that they improve.

3.1.1 Partial Optimization Attempts

This line of work does not consider the interplay between statistical utility and system utility. Instead, they mainly focus on lifting either utility while leaving the other one ignored or controlled to a limited extent.

Statistics-Oriented. To approach the convergence rate in centralized settings where the data is IID, CSFedAvg [17] advocates that clients with a lower degree of non-IID data should participate more often. To this end, the authors propose weight divergence to capture the non-IID degree of data owned by a client. More precisely, it measures the normalized Euclidean distance between a client's model and the reference model trained by the server with auxiliary IID data. According to the 500-client simulation over CIFAR-10 and Fashion MNIST, CSFedAvg reduces the time-to-accuracy by up to $4.0\times$ and $2.7\times$, respectively, compared to random selection.

Besides heuristic methods, some researchers use reinforcement learning (RL) algorithms to learn which clients to select in the presence of data heterogeneity. For example, FAVOR [19] seeks to reduce the number of rounds to reach a target accuracy with a deep Q-learning network (DQN) [105]. To capture each client's statistical characteristics, it takes the low-dimension representations of local models as the RL states. Compared to random selection, FAVOR can reduce the communication rounds by up to 49% in three image classification tasks. On the other hand, the training overhead for the RL agent may become an obstacle to FAVOR's real-world applications. Specifically, it is reported to take more than 100 episodes to train an agent suited for a specific learning task, which could be prohibitive as each episode corresponds to an entire FL process, i.e., training a global model from scratch for reaching a target accuracy.

System-Oriented. In synchronous training, clients with the lowest system utility bottleneck the speed of a federation round. A straightforward way to bound the time usage is setting a deadline for randomly selected clients' to report updates and ignoring any update submitted after the deadline. To avoid waste of computing resources, FedCS [18] takes a step further by proactively selecting a set of clients whose participation is not likely to miss the deadline according to latency estimation results. As there can be multiple eligible sets, FedCS further favors the solution with the largest scale of participation, which reduces part of the loss in statistical utility. Technically, the whole problem is formalized as a complex combinatorial optimization, and the authors resort to a greedy algorithm for efficient approximation. As indicated in their 1000-client simulation, FedCS outperforms FedLim (modified FedAvg with per-round deadlines imposed) by up to $1.2\times$ and $1.8\times$ in the time-to-accuracy when training over the non-IID CIFAR-10 and Fashion MNIST datasets, respectively.

3.1.2 Co-Optimizing Statistical/System Utility

Given the coupled nature of clients' system utility and statistical utility, it is more desirable to navigate the sweet point of jointly maximizing both of them.

Coarse-Grained. TiFL [20] first considers increasing the system utility. To that end, it divides clients into different tiers based on the observed runtime performance, and at each round only selects clients from the same tier for mitigating the waste of resources due to idle waiting for stragglers. To reduce the average iteration span, it also limits the number of times a (slow) tier can be selected. On top of that, the statistical utility is respected by prioritizing tiers with lower testing accuracy whenever there is more than one electable tier. Compared with FedCS, TiFL bears some resemblance in limiting the participation of less capable clients, while being more aware of the statistical utility. As reported in a 50-client cluster with 5 client tiers, TiFL achieves an improvement over random selection by up to $3\times$ speedup in overall training time and by 6% in accuracy.

Fine-Grained. Compared to TiFL, Oort [21] reconciles the demand for enhancing both system utility and statistical utility in finer granularity. Specifically, it associates each client with a continuous score and prioritizes those clients with higher scores. The score is meant to be a principled measurement of both the statistical utility (determined by the training loss) and the system utility (estimated from historical response latency). As some components of the score cannot be known in advance until the corresponding client's first participation, or cannot be guaranteed to be stable due to the client's runtime dynamics, the score estimation process is modeled as a Multi-Armed Bandit (MAB) problem and tackled by the Upper Confidence Bound (UCB) algorithm [106]. Apart from the scoring backbone, Oort also aims to address other practical issues like staleness and robustness. Oort was evaluated on a 1300-client GPU cluster with realistic datasets and simulation on the client heterogeneity. Compared to random selection, it reduces the training time by up to $14\times$ and improves model accuracy by up to 9.8%.

Besides the UCB algorithm, researchers also experiment with more sophisticated RL methods to cherrypick participants. Notably, AutoFL [22] learns to select participants and execution targets for each individual based on the Q-Learning algorithm [107]. To achieve efficient FL execution, it identifies RL states that are critical to energy efficiency, convergence time, and accuracy. It also defines RL rewards that track the energy consumption of clients and model accuracy. Compared to random selection, AutoFL is reported to improve the average FL energy efficiency by up to $4.3\times$, while also exhibiting better training accuracy. Similar to FAVOR (mentioned in §3.1.1), however, training the Q-Learning model to converge needs multiple episodes, each of which corresponds to an entire FL training process. Its practicality could thus be challenged whenever offline training is infeasible or costly.

3.2 Optimizing the Configuration Phase

In the configuration phase, there are mainly two processes that are responsible for the time-to-accuracy performance. One is downlink (i.e., server-to-client) model transmission and the other is local model training. Thus, both aspects can be investigated for system optimization. As for communication overhead reduction, one can reduce the size of model updates (§3.2.1) and decrease the synchronization frequency (§3.2.2). To lower computational overhead, one can shorten the training latency by balancing the workload across clients (§3.2.3), as well as reducing the number of rounds taken to converge by adopting heterogeneity-aware training algorithms (§3.2.4). As the uplink model submission (i.e., client-to-server) that takes place in the reporting phase shares the same operational space as the downlink one, we combine the discussion on both of them in this section for brevity.

3.2.1 Model Update Size Reduction

Prior arts of model update size reduction mainly fall into three camps: quantization, sketching, and sparsification. Ahead of the emergence of FL, the exploration of these directions has already been initiated in the context of traditional distributed learning. While their communication merits are mostly reproducible in FL, they also face new challenges due to the privacy regulations and client heterogeneity, which we will also point out hereafter.

Quantization. Quantization converts each scalar in a model update to its low-bit representation which takes up less space. While quantization has already gained its fame in traditional distributed learning and we refer the readers to dedicated surveys like [108] for more details, here we only introduce the most representative work. As the first quantization work in model training with rigorous convergence proof, QSGD [23] performs unbiased quantization with standard random dithering, a technique borrowed from image processing. Since its birth, related works have been emerging with more aggressive quantization bit-widths and more appealing empirical performance. For example, TernGrad [24] advocates using only ternary values $(0, \pm1)$ in the uplink direction, while signSGD [25] can use only binary signs $(\pm)$ in both uplink and downlink communication. It is worth mentioning that a popular technique in tackling the precision loss brought by quantization is error feedback, whose basic idea is to accumulate the previous quantization errors and compensate for them in the current round. Leveraging this technique, ECQ-SGD [26] performs consistently better than QSD in terms of both convergence speed and accuracy, while EF-SGD [27] also achieves a narrower generalization gap from centralized training compared to signSGD.

Despite their generality, there are some practical concerns about applying these general quantization strategies to FL due to privacy constraints and client heterogeneity. For example, determining the clipping threshold for quantization needs to exploit the knowledge about its inputs (i.e., local model updates) for reducing the induced error as in dACIQ [109]. However, an FL client does neither possess a priori knowledge of others' model updates nor should it request these sensitive values. To work out a globally applicable clipping threshold, we may need to share some less sensitive information (e.g., the maximum and minimum values in local updates) across clients for threshold estimation as in BatchCrypt [60]. Still, whether such a circumvention guarantees robust estimation and immunity to privacy attacks remains an open question.

Sketching. Existing quantization approaches assume the input values follow a certain distribution (e.g., uniform or bell-shaped), which may not always be the case in model updates [28]. To be more general, some researchers introduce sketching methods where some memory-saving data structures are used to approximate the exact distribution of model update values in a single processing pass over the values. For example, SketchML [28] utilizes a quantile sketch method to generate a non-uniform mapping from gradient values to low-bit integers. SketchML achieves empirical success such as decreasing the gradient size by around $7\times$ and is the first effort to introduce sketching for compressing model updates in ML training. Similar to quantization, sketch algorithms can also make use of error feedback techniques to efficiently amend the errors induced by the approximation, as demonstrated in SketchedSGD [29] and FetchSGD [30]. There is also sketching practice that compresses auxiliary variables apart from model updates, such as sketching clients' momenta and per-coordinate learning rates as in [31].

Sparsification. While quantization and sketching operate at the precision level in terms of model size reduction, sparsification operates at the coordinate level. Specifically, sparsification allows each client to transmit only a sparse subset of its model updates, while the rest are accumulated and incorporated into future training. Technically, the sparsified gradient is obtained by first performing element-wise multiplication on the original gradient with a certain 0/1 mask and then discarding zero elements. The mask is typically randomly generated as in [32], while another commonly used variant is the top $s\%$ scheme where 1 is given to the coordinates that rank top $s\%$ in absolute magnitude and 0 otherwise [33], [34], [35], [36]. The top $s\%$ method is reported to reduce network traffic by up to three orders of magnitude, while still preserving model quality [33], [34].

While similar cost savings are transferable to plaintext FL, it is unclear whether sparsification can be further compatible with cryptographic techniques that are widely adopted for privacy enforcement in FL. For example, apart from the uplink model updates, it is also desirable to sparsify the downlink global update for fully releasing the potential for communication improvement. However, implementing the downlink sparsification may not be feasible when the server is not aware of the plaintext values of the aggregated update as a result of the applied Secure Multi-Party Computation (SMPC) [59], [78], [110], [111] or Homomorphic Encryption (HE) [60], [112], [113], [114] techniques.

It is noteworthy that as quantization (or sketching) and sparsification are orthogonal to each other, they can be combined to reap the most benefits in terms of model size reduction [115], [116].

3.2.2 Synchronization Frequency Reduction

At its core, the reduction in the synchronization frequency is achieved by identifying and precluding redundant synchronization efforts. This can be operated at different granularities ranging from clients, layers to individual parameters in a model update.

Client-Level. The importance of an entire model update is usually measured by some numerical features. In the most intuitive form, a model update in Gaia [37] is considered significant if its magnitude relative to the current value $|\frac{Update}{Value}|$ exceeds a specific threshold such as 1%. While the magnitude may serve as a good indicator for how data center learning performs, it does not work in FL where determining an appropriate threshold is hard due to clients' heterogeneity. As such, some researchers propose to involve the comparison with some reference points for more robust measurement of the importance. For example, [38] tracks the Euclidean distance between the local model and a reference model, while CMFL [39] focuses on the number of coordinates with the same sign in the local model and the most recent global model.

Layer-Level. Apart from considering a model update as a whole, another line of work tries to reduce the synchronization frequency on a layer basis. A representative work done in this direction is TWAFL [40] where the model aggregation is conducted layer-wise. As observations made in deep neural network (DNN) fine-tuning [117], shallow layers in a DNN learn general features across different datasets while deep layers learn ad hoc ones. TWAFL hence proposes to update shallow layers more frequently than deep ones as they are more responsible for the overall quality of the global model.

Parameter-Level. Some industrial practitioners also consider whether to synchronize for each round at the level of individual parameters. Noticing that each parameter usually evolves in a transient-then-stable manner, i.e., it first varies drastically and then settles down around a certain value with slight oscillation, APF [41] proposes to stop synchronizing those parameters whose evolution has reached a stationary phase.

3.2.3 Training Latency Reduction

A client's training latency is determined by both its computational workload and resource capabilities. While the latter cannot be altered, the former still leaves room for optimization innovations. We discuss one major line of such efforts.

Load Balancing. Given the variations in computing power and data volume, clients may not finish the training process at the same time. To mitigate the resulting straggler effects, [42], [43] suggest balancing the amount of training data across clients. Specifically, they turn to RL techniques for determining the optimal number of data units used in a communication round for each participant, intending to minimize the time and energy consumption and maximize the volume of involved data. While these solutions can reduce the round latency, the number of rounds needed for convergence does not necessarily remain unchanged because they do not account for data heterogeneity and thus the contribution of slow clients with important data could be throttled. To jointly consider heterogeneous system speed and data distribution during load balancing, [44] carefully formulates an optimization problem where each client is associated with a weight that diminishes exponentially with the disparity between its number of labels and that of the population. Compared to even data allocation, this work manages to reduce the end-to-end time-to-accuracy.

Load balancing can also be achieved by varying the number of optimization steps or the complexity of local models. For example, FedProx [45] balances the system load across clients by formulating an inexact learning problem that allows variable steps of local solvers. On the other hand, HeteroFL [46] assigns sub-models with different widths of hidden channels to clients so that clients with fewer capabilities can train smaller sub-models. All sub-models share the same model architecture, and thus normal model aggregation is still possible. The authors empirically show that the quality of the global model trained with heterogeneous sub-models is comparable to that trained with full-sized local models.

Apart from the computational load, it is sometimes also beneficial to balance the communication load across clients, especially when the network conditions are complicated as in wireless connections. For example, targeting a mobile edge computing (MEC) scenario where Time Division Multiple Access (TDMA) is implemented, [47] optimizes both the data batch size and uplink/downlink frame time slots for each client to achieve the maximum learning efficiency. In addition to coping with CPU computing, the authors further extend the optimization problem to the scenario where devices are equipped with GPUs for training.

3.2.4 Training Round Reduction

In the settings with heterogeneous data, more local computation in a communication round does not necessarily lead to fewer numbers of rounds for reaching satisfying optima [55]. Thus, adopting heterogeneity-aware techniques such as adaptive optimization and bias reduction can help guarantee the convergence speed in FL practice.

Optimizer State Synchronization. It is common practice for first-order moment optimization to apply momentum to dampen oscillations [118], [119]. However, in the federated settings, if the clients' momenta are separately updated, they may deviate from each other due to non-IID data distributions. Thus, there are researchers proposing optimizer state synchronization frameworks where clients' optimizer states are synchronized by the server periodically. PR-SGD-Momentum [48] is aligned with this direction and also gives proof of the linear speedup of convergence w.r.t. the number of workers. FedAC [49] also applies momentum at clients with periodic synchronization, while it is proven to obtain the same linear speedup property with asymptotically fewer rounds of synchronization. MFL [50] is another similar idea with theoretical guarantees, but it focuses on accelerating deterministic gradient descent (DGD) instead of SGD, unlike the previous two studies.

Client Bias Reduction. Due to data heterogeneity, clients' model updates can be biased towards the minima of local objectives, known as "client drift" in the literature [120], which hinders the convergence of the global model. To reduce the variance across clients, a natural idea is to regularize local objective functions for minimizing the drift. For example, assuming bounded dissimilarity between local functions, FedProx [45] limits the Euclidean distance between local models and the global one by regularizing the square of the distance. FedDANE [51] uses the same regularization term, while it formulates the other part of the objective

function following the Distributed Approximate NEwton (DANE) method [121] in classic distributed optimization. Despite having a similar theoretical convergence rate as Fed-Prox, FedDANE underperforms FedProx in practice where the data heterogeneity is high and the participation rate is low, suggesting a discrepancy between theory and practice which needs further investigation. The objective function in FedDyn [52] also considers the combination of a linear term and an L2 regularizer, with a different linear term which is formulated to align the empirical loss surfaces of clients. In theory, FedDyn ensures that the consensus point of model convergence across clients will be consistent with the global stationary solution as long as local models converge, regardless of the degree of data heterogeneity.

Apart from regularizing local objectives, variants in clients' local updates can also be reduced by leveraging the idea of control variates borrowed from the convex optimization literature. For example, in SCAFFOLD [53], each of the clients and the server maintains a control variate, and at each local step, a client de-biases its local updates with two control variates: one of its own and the other broadcast by the server. SCAFFOLD converges provably faster than FedAvg [6] without any assumption made on the client selection or data heterogeneity. MIME [54] considers a similar idea but makes a different choice on the specific definition of control variates.

While the use of control variates requires persistent client states, there exists another line of work that works for stateless clients: posterior averaging. Instead of approaching FL as optimization, this line of work formulates the problem as a posterior inference one. Compared to traditional federated optimization, posterior inference can benefit from an increased amount of local computation without risking stagnating at inferior optima. FedPA [55] instantiates this idea with an efficient algorithm to conduct federated posterior inference with linear computation and communication costs.

3.3 Optimizing the Reporting Phase

In this phase, the operation room for system optimization is limited to either model uploading or model aggregation. As the former has already been combined in the last section, we hereafter focus on optimizing the aggregation process with two main directions explored in the literature: (1) directly reducing the aggregation latency at each round (§3.3.1), and (2) expediting the convergence rate in the long run through conducting adaptive aggregation (§3.3.2).

3.3.1 Aggregation Latency Reduction

Compared to local training in the configuration phase, model aggregation involves less intensive computation. However, its latency can still be salient because (1) large-scale participation can put pressure on the communication, and (2) the deployment of security methods can complicate the computation. We hereafter introduce the respective optimization efforts in the literature.

Hierarchical Aggregation. The downsides of the traditional two-layer (server-client) system include (1) instability: the network bandwidth may be slow or unpredictable, especially in public networks and/or under geo-distributed

settings; (2) risk of scalability: the cloud server may suffer from network congestion when concurrently receiving too many local updates; (3) heterogeneity: the straggler effects could be exacerbated by imbalanced network bandwidth.

To address these issues, some researchers resort to a hierarchical design of model aggregation by introducing an extra level of edge servers, each of which is typically responsible for a small number of clients with proximity. For instance, in HierFAVG [56], after a fixed number of local updates on clients, each edge server aggregates its own clients' models. Subsequently, after another fixed interval of edge aggregation, the cloud server aggregates all the edge servers' models. It is proven that HierFAVG still guarantees convergence, and empirical studies with synthetic FL datasets show that it reduces the time-to-accuracy by up to $2.7\times$ in a simulated cloud-edge-client environment. A concurrent work HFL [57] also considers a similar design, while it does not attach theoretical analysis on its convergence behaviors. HybridFL [58] further extends this primary design with two ideas: (1) quota-triggered edge-level aggregation: edge nodes stop waiting for more local updates once receiving a sufficient number of them; and (2) immediate cloud aggregation: cloud-level aggregation is conducted right after the edge-level one is completed. This decouples each pair of interactions (i.e., cloud-edge and edge-client), thereby further mitigating the impact of client drop-out and stragglers.

Despite changing the aggregation rules, the hierarchical designs mentioned above still focus on establishing the convergence to a single global model, which does not deviate from the learning paradigm discussed throughout this survey. On the other hand, there also exist other formulations of collaborative learning such as clustered FL where clients are assigned to different groups and aggregation takes place within a group [122], [123], [124]. As they aim to build personalized models for each group of clients, they are considered paradigm innovations to standard FL and thus fall out of the scope of this survey (§2.2).

Lightweight Privacy-Preserving Aggregation. As mentioned in §2.3.1, uploading model updates in the clear may be vulnerable to exploratory attacks which plague clients' privacy. Therefore, model aggregation is preferably safeguarded by cryptographic techniques, which inevitably induces extra computation and communication overhead.

One line of this work leverages secure multiparty computation (SMPC). One of the most impactful works is Google's secure aggregation (SecAgg) protocol, where pseudorandom masks are used for data confidentiality and Shamir's secret sharing scheme [125] is used to accommodate client dropout [78]. It has provable rigorous privacy guarantees under both passive and active adversary models. Given its quadratic communication overhead during secret sharing, SecAgg has inspired many other SMPC protocols with improved performance. For example, SecAgg+ [126] optimizes SecAgg in both communication and computation by replacing the complete graph with a random sparse one of logarithmic degree. TurboAgg [59], on the other hand, attempts to optimize SecAgg through the use of multi-group circular topology, additive secret sharing, and Lagrange coding. Compared to SecAgg+, TurboAgg incurs

a slightly higher overhead in the asymptotic sense for both communication and computation, and only focuses on tackling honest-but-curious adversaries. While all of the above schemes rely on Shamir's scheme for sharing clients' secrets, there also exist other secret sharing techniques that offer different trade-offs across performance, privacy, and dropout resilience, e.g., encoding using Maximum Distance Separable matrices as in LightSecAgg [127], or Fast Fourier Transform as in FastSecAgg [128].

Besides SA, Homomorphic Encryption (HE) is another commonly used privacy-preserving aggregation technique that comes with prohibitively high message inflation and runtime overhead. In response, BatchCrypt [60] implements an end-to-end solution for batching multiple plaintexts into one large plaintext so that HE-related operations can be performed in a data-parallel manner. BatchCrypt is shown to speed up the training by $23\times$-$93\times$ compared to plain Paillier [129] (a prevalent variant of HEs), but it still leaves the message inflation suboptimal and is incompatible with top s% sparsification approaches [61]. Instead of optimizing traditional HE schemes by batching, FLASHE [61] proposes a lightweight HE scheme that is tailored for cross-silo FL. It induces negligible ($\leq 6\%$) computational overhead and no network communication overhead compared to plaintext FL for staying symmetric. Its performance is also optimized when interacting with sparsification techniques.

3.3.2 Adaptive Aggregation

In FedAvg [6], the de facto standard aggregation method, local model updates simply get weighted by the corresponding numbers of training samples and then added up. While it guarantees convergence when even dealing with non-convex empirical risk functions in IID data settings [53], [130], it is observed to yield unstable convergence behavior or even divergence when it faces models trained with arbitrarily non-IID data. There are thus rising interests on whether the aggregation can be more adaptive w.r.t. the data heterogeneity across clients.

Server-Side Optimizers. Other than accelerating convergence with local momentum (§3.2.4), there are also exploration efforts on server-side momentum. As there is originally no optimizer at the server in FL, these methods first need to generalize the existing aggregation algorithm. Specifically, at each round, instead of collecting local model weights, the server instead collects their changes and treats these changes as the "pseudo-gradient" for the server, which the server can use to update the global model with adaptive optimizers. FedAvgM [62] initiates the empirical studies with the simplest form of momentum applied at the server, while SlowMo [63] independently proposes a similar scheme and also attaches the theoretical analysis for its convergence behaviors. A recent work [64] makes more sophisticated use of momentum such as adopting AdaGrad [131], Adam [132] and YOGI [133] optimizers (which correspond to FedAdaGrad, FedAdam, and FedYOGI, respectively). It is shown that FedYOGI consistently outperforms FedAvgM in terms of validation performance for both sparse- and dense-gradient FL tasks. Server-side momentum methods feature no need for persistent states or computation burdens at the client end, making them well suited to cross-device scenarios.
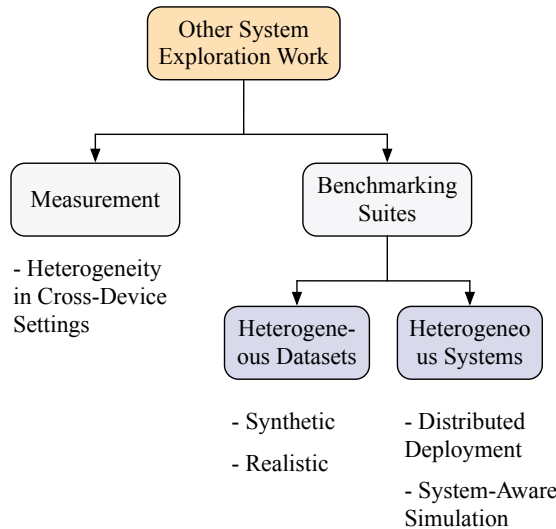
Fig. 4: Taxonomy of the work introduced in §4.

4 MEASUREMENT AND BENCHMARKING TOOLS

Aside from innovating optimization solutions, there are also researchers contributing with cornerstone works that benefit the community with informative insights from systematical measurement studies (§4.1) and grounding benchmarking tools (§4.2), as visualized in Fig. 4.

4.1 Measurement-Based Research

Due to the complicated interplay between statistical utility and system utility, there are a few measurement studies which dedicate to conducting thorough investigations and providing actionable implications for interested researchers.

- *FLASH*[2] [65]: FLASH particularly studies the impacts that heterogeneity has on both the statistical utility and system utility. This work contributes to the community mainly with three aspects: (1) it establishes a novel large-scale dataset that reflects the system and state heterogeneity among 136,000 real-world smartphones; (2) it implements and open-sources an FL platform that provides a heterogeneity-aware environment for experiments; (3) it reports comprehensive findings that stem from the authors' extensive measurements atop the platform.

  Among the authors' numerous implications, there are two fresh findings at the time when the study is written. First, gradient compression methods (e.g., Gradient Dropping [33] and SignSGD [25]) can hardly shorten the convergence time under heterogeneous cross-device settings. Second, advanced aggregation algorithms that overlook some aspects of heterogeneity will be less effective in realistic settings. Readers can refer to the paper for more details.

4.2 Benchmarking Suites

Realistic benchmark suites are necessary for enabling fair, insightful, and reproducible evaluation of the effectiveness

---

2. We name it after the corresponding Github repository handle [134].

of system optimizations. As the time-to-accuracy performance relies on both the statistical utility and system utility (§2.2), in the following summary of existing benchmarking tools, we aim to cover diverse aspects of simulating practical FL: data characteristics, client capabilities, and availability.

4.2.1 Training Datasets

There are two prevalent categories of training datasets. One line of work is *derived from conventional ML datasets* (e.g., CIFAR [135], MNIST [136], and Fashion-MNIST [137]). To synthesize the non-IID nature as in real FL scenarios, the data partitions in these datasets are typically formed by restricting the number of data classes each client has (e.g., partitioning by shard-based methods as in [6] or latent Dirichlet allocation (LDA) processes as in [52], [55], [62], [64]). Although the data generated in such a way are indeed non-IID, they may not perfectly represent the real-world characteristics. For instance, besides the label distribution skew, in reality, non-IID data may also involve feature distribution skew (e.g., same words with different stroke widths), same labels with different features (e.g., images of clothing vary due to regional differences) and same features with different labels (e.g., the same context mapped to different next words due to personal habits) [73], [104].

In contrast, the other type of datasets is *collected in real distributed scenarios* and thus naturally captures the FL features. We briefly introduce existing open-source attempts in curating such datasets as follows.

- *LEAF* [66]: LEAF is an actively maintained project, which currently consists of 6 datasets spanning multiple applications such as computer vision (CV) (FEMNIST and CelebA) and natural language processing (NLP) (Shakespeare, Reddit, and Sentiment140). Each dataset is generally formed by splitting the corresponding public dataset by the original contributors of the data samples. In other words, the non-IID nature comes from the unique behavior style of each contributor.

- *FedScale* [68]: Similar to LEAF, FedScale also collects realistic datasets and partitions them by unique client identification. FedScale currently includes 18 datasets (including iNature, OpenImage, Google Landmark etc.) which span 10 FL tasks. Apart from the comprehensive coverage of tasks, FedScale has further made four contributions to the community: (1) it has the training, validation, and testing set well established; (2) it streamlines different datasets into a unified format; (3) it accounts for various participation scale from hundreds to millions of clients; and (4) it provides handy APIs for users to customize their datasets.

- *OARF* [67]: for a specific FL task, OARF assembles real-world datasets from different sources to realize data heterogeneity. For example, for sentiment analysis, it combines both the IMDB Movie Review and Amazon Moview Review datasets. In training, datasets belonging to different sources are distributed to different parties. As such, the data are split in a dataset-wise manner instead of a sample-wise one. OARF currently covers 9 tasks in CV and NLP.

Besides the above systematic collection, there are also other separately maintained realistic datasets, such as `Stackoverflow` [138] and `PERSONA-CHAT` [139].

4.2.2 Production Systems and Simulation Platforms

In addition to setting up data heterogeneity, we also need to incorporate system heterogeneity in realistic benchmarking. The most straightforward way to study FL designs with system utility borne in mind is to deploy in *production-oriented systems*. Such systems not only embed the ML backbone but also address practical problems like authentication, communication, encryption, and deployment to physical distributed environments. We sketch the open-source representatives.

- *FATE* [69]: FATE is an FL framework that can be deployed in distributed environments. In addition to its flexible ML pipeline, FATE also features several aspects that further facilitate the research on various goals of practical FL: (1) it supports privacy-preserving computation by implementing cryptographic algorithms such as the Diffie-Hellman key agreement [140] and homomorphic encryption [129]; (2) it covers different training architectures including horizontal FL, vertical FL, and federated transfer learning; and (3) it allows a certain degree of customization on the FL pipeline such as the aggregation step. Given its heaviness in resource consumption, FATE is preferable in powering cross-silo applications instead of cross-device ones.
- *FedML* [70]: FedML is also a secure and versatile FL framework that supports distributed mode. Compared to FATE, FedML is more flexible in communication engineering due to the ease of customizing message flow and topology definitions. It is also more lightweight and can thus accommodate training on mobile or IoT devices. Moreover, it can be accelerated with GPUs, while FATE is currently not compatible with hardware accelerators.
- *Flower* [71]: Flower is a concurrent work with FedML, and it concentrates on providing a unified approach for FL with mobile devices. Similar to FedML, Flower bears in mind the goals of being (1) lightweight, (2) extensible, (3) scalable, and (4) compatible with diverse mobile platforms (e.g., Android and iOS) and ML-frameworks (e.g., PyTorch [141] and Tensorflow [142]). It also implements secure aggregation methods like SecAgg (c.f. §3.3.1) with easily configurable APIs [143].
- *Plato* [72]: similar to Flower, Plato also aims to facilitate FL research atop multiple ML backends including PyTorch, Tensorflow, and MindSpore [144]. Different from its counterparts, Plato is optimized in the development process, e.g., making extensive use of plugin mechanisms to maximize extensibility and maintainability. Apart from lightweight distributed deployment, Plato also offers solutions to integrate secure communication, reverse proxy, and load balancing for best fitting in production environments.

Although using production systems yields the most realistic insights, it may not be practical for researchers with limited resources and time budgets. To meet the growing demand for conducting agile FL research, several platforms *that enable system-aware simulation* have been developed. As opposed to system-unaware simulators (e.g., Tensorflow Federated [145], PySyft [146], LEAF [66], OARF [67], and FedEval [147]), these platforms respect the impact of client system heterogeneity by associating each client with her computation and communication speed, as well as availability dynamics, which are either set manually by developers or by replaying realistic traces. In addition, these platforms also excel in producing comprehensive metrics needed in performance analysis. Compared to real deployment, on the other hand, these systems allow researchers to make fast-forward progress without being blocked by real-world bottlenecks in computation and communication.

- *Flower* [71]: besides deployment on real mobile devices as just mentioned, Flower also supports simulation in the cloud with configurable system-level parameters such as bandwidth constraints and computational capabilities. With that, researchers can experiment with larger and more compute-intensive FL workloads that cannot be run on today's mobile devices.
- *FedScale* [68]: aside from curating real-world datasets (§4.2.1), FedScale also builds an automated runtime to simulate FL in realistic settings. By design, FedScale integrates `AI Benchmark` and `MobiPerf Measurements` system traces to simulate clients' heterogeneous training speed and network throughput, respectively. It also incorporates a large-scale user behavior dataset that was formulated in [65] to emulate clients' availability dynamics. Compared to Flower, it lacks support for deployment on real distributed devices. Still, it broadly simulates realistic cross-device heterogeneity and can embrace new behavior traces with its APIs.

5 RELATED WORK AND CONCLUDING REMARKS

5.1 Related Surveys

The motivation for this survey stems from three observations. First, few in-depth surveys focus on system optimization for FL. As FL features strict compliance to privacy regulations as opposed to traditional distributed learning, many survey efforts are directed to the unique challenges such as enforcing data privacy and model robustness [148], [149], [150], [151], [152], while the system optimization issues receive less attention in dedicated surveys.

As for common system issues shared by FL and traditional distributed learning, there do exist extensive surveys with detailed discussion. In [153], the authors discuss the realm of mobile distributed machine learning, where algorithms are classified into three categories: 1) machine learning optimizers, 2) distributed optimization algorithms, and 3) data aggregation methods. In [108], the authors provide a detailed survey of communication-efficient distributed training algorithms. They identify four key factors that affect the communication cost in distributed learning and organize the literature accordingly: 1) synchronous scheme, 2) system architecture, 3) compression techniques

and 4) parallelism of communication and computation. [154] also studies the communication issues in distributed deep learning, however, with a focus on the commonly used lossless methods. It also contains a quantitative analysis of these methods based on real-world experiments. Compared to our survey, the scope of these surveys is not narrowed down to FL and thus does not fully capture all the system optimization challenges that are unique in FL. Notably, FL has in its standard workflow the selection phase which needs particular investigation due to client heterogeneity, while traditional distributed learning does not even have the notion of clients.

We notice that there are comprehensive surveys that cover a wide range of FL techniques including system-level optimizations. In [73], the authors discuss in-depth the advances and open problems in FL at the time of writing. The presented problems and solutions are presented in four categories: 1) efficiency and effectiveness, 2) privacy preservation, 3) robustness enforcement and 4) fairness establishment. In [155], the authors extensively introduce the challenges and research directions of FL at and for mobile edge network. As for FL at mobile edge network, they particularly elaborate techniques on 1) communication efficiency, 2) resource allocation, and 3) privacy and security. In [120], the authors provide recommendations on algorithm-level optimization of FL applications, while they also briefly sketch the system-level constraints and practices. They discuss the reduction of communication, computation, and memory costs, as well as propose a basic model to estimate the communication efficiency of cross-device FL deployment. In [156], the authors present a fine-grained multi-level classification of the FL literature, spanning six major topics: 1) statistical challenges, 2) communication efficiency, 3) client selection and scheduling, 4) security concerns, 5) privacy concerns, and 6) service pricing. Compared to this survey, the above-mentioned work is neither intended to focus on system-level optimizations nor do they categorize the literature by the phases in the synchronous training process. The survey most closely related to ours is [157] which is dedicated to summarizing system-level optimizations used in asynchronous FL. As their studied architecture is fundamentally different from ours, their survey could be a good complement to this survey. It is also noteworthy that their classification mechanism on existing techniques is built upon the type of client heterogeneity, which also differs from our taxonomy.

In short, this survey aims to provide a succinct yet complete view of an essential domain in the FL literature: system optimization in synchronous federated training. We also summarize in Tab. 1 the main similarities and differences between our survey and the existing surveys that are closely related.

5.2 Future Research Directions

The primary goal of this survey is to help researchers design future optimization solutions. To stimulate more directives for FL practitioners, we discuss in the following some possible future directions that we derive from the literature as well as our development practice.

5.2.1 On the Selection Phase

While RL agents are commonly used in guiding client selection for their beneficial balance between exploitation and exploration [19], [22], their training costs are inherently high. Before an RL agent is ready for use, it has to learn from tens to hundreds of full-sized FL training processes, whose cost may not be justifiable in practice. Inspired by the experience from container management [158], we are interested in whether certain techniques, e.g., transfer learning [159], can be adopted so that an RL agent can be trained from prior FL tasks, instead of learning from scratch for every single task.

Moreover, existing algorithms all assume the participation scale (i.e., the number of clients training simultaneously) to be the magnitude of a hundred, because involving more clients in a round is observed to have marginal benefits under primary aggregation methods (e.g., FedAvg [6]). On the other hand, the number of available clients at each minute can be as many as thousands in the cross-device practice. It is still desirable to explore the selection of a larger crowd for boosting the time-to-accuracy performance.

5.2.2 On the Configuration Phase

Aligned with the observations in the literature [68], most prior arts consistently configure different clients. Although there exist some heterogeneity-aware efforts like load balancing where the number of batches, batch size, and number of local epochs can vary across clients (§3.2.3), we anticipate that the design space for heterogeneity-aware client configurations could be larger, e.g., using different compression ratios or synchronization frequencies.

To achieve efficient and secure communication, it is also worth studying how to combine sparsification techniques with privacy-preserving methods. For example, model sparsification is by nature not compatible with secure multiparty computation protocols such as SecAgg [78], because the coordinates of sparsified model values often vary across clients, preventing the pairwise masks from being canceled out during model aggregation as required by SecAgg. A recent work named SparseSecAgg [160] attempts to tackle this issue, however, it implies the need for sharing some sparsified locations between each pair of clients, which cannot be directly extended to conventional sparsification schemes such as the top $s\%$ scheme (§3.2.1). More general reconciliation between the two techniques is still ripe for future investigation.

5.2.3 On the Reporting Phase

As the system bottleneck is usually assumed to locate in clients instead of the server, most of the existing optimization efforts focus on improving the utility (system and statistical) of clients. It is thus interesting to investigate whether such an assumption holds in practice, especially when the scalability of the server is restricted due to rigid capabilities or limited budgets, or when its aggregation latency is not negligible due to the presence of large-scale models.

Besides, existing lightweight privacy-preserving aggregation methods are not compatible with robustness enforcement techniques [73]. Because the true values of local models are concealed from the server, it has no way of inspecting their numerical features [161], [162], [163], [164]

TABLE 1: Comparative summary between our survey and the most related work.

Survey	Similarities	Differences
Gu et al. [153]	Similar to our survey, this survey addresses some common challenges of synchronous federated training such as communication efficiency.	• As for machine learning optimizers, it does not discuss the server-side ones which are FL-specific. • Our work addresses additional challenges that are not mentioned in this survey such as participant selection and aggregation efficiency.
Tang et al. [108]	Similar to our survey, this survey addresses some common challenges of synchronous federated training such as communication efficiency.	• It considers distributed vanilla SGD, which is different in convergence characteristics from FL, a special case of local SGD. • Our work addresses additional challenges that are not mentioned in this survey such as participant selection and client bias reduction.
Shi et al. [154]	Similar to our survey, this survey addresses some common challenges of synchronous federated training such as communication efficiency.	• It focuses on controlling the system architecture and scheduling algorithms in data center learning, which is not applicable to synchronuous FL. • Our work addresses additional challenges that are not mentioned in this survey such as participant selection and data heterogeneity.
Kairouz et al. [73]	Similar to our survey, this survey addresses some common challenges of synchronous federated training such as client bias reduction and communication efficiency.	• The scope of this survey is quite board, where for example theoretical analysis of convergence and privacy-preserving mechanisms are also extensively discussed. • Our work addresses additional challenges that are not mentioned in this survey such as participant selection and load balancing.
Lim et al. [155]	Similar to our survey, this survey addresses some common challenges of FL such as communication efficiency and participant selection. It also mentions the standard workflow of synchronous training.	• The scope of this survey is quite board, where many applications of FL in mobile edge network, e.g., cyberattack detection, are also elaborated. • They do not organize system-level optimizations by the phases in the standard training workflow. • Our work addresses additional challenges that are not mentioned in this survey such as client bias reduction and lightweight privacy-preserving aggregation.
Wang et al. [120]	Similar to our survey, this survey addresses some common challenges of FL such as communication efficiency and lightweight privacy-preserving aggregation.	• It centers on the algorithm-level optimizations as well as their theoretical guanrantees, while the challenges and directives of system-level issues are briefly introduced. • Our work addresses additional challenges that are not mentioned in this survey such as participant selection and load balancing.
Wahab et al. [156]	Similar to our survey, this survey addresses some common challenges of FL such as participant selection and communication efficiency.	• The scope of this survey is quite board, where even training service pricing are also included. • Our work contains additional exploratory work in the system community that is not mentioned in this survey including measurement studies and benchmarking suites.
Xu et al. [157]	Similar to our survey, this survey focuses on optimizing FL training process in the precence of client heterogeneity.	• Its target architecture is asynchornous and thus the intersection of its introduced techniques and ours are fairly small. • Our work sorts the literature by the phases in the standard synchronous training workflow, which is distinct from the classification mechanism used in this survey.

or validating their performance for anomaly detection [81], [165], [166]. This raises the tension between privacy and robustness. Leveraging Trusted Execution Environment (TEE) [167], [168] to perform model inspection securely might be a helpful workaround. However, due to current limits on TEE's hardware capabilities [169], there could be a foreseeably large performance downgrade of doing so. It thus remains largely unexplored as to how to navigate the sweet point of jointly maximizing accuracy, performance, privacy, and robustness in synchronous federated training.

5.3 Discussion

We further discuss the applicability of the above-introduced optimization techniques in different contexts.

5.3.1 Cross-Device FL and Cross-Silo FL

FL applications are often categorized as either cross-device FL (where the participants are a mass of less capable mobile or IoT devices) or cross-silo FL (where the participants are 2-100 organizational entities) [73]. While the FL workflow that we base on throughout this survey is primarily proposed for cross-device FL [65], [68], [74], it also generalizes to cross-silo settings. Hence, the scope of this survey does not preclude cross-silo FL, and hence many practical methods mentioned here should apply to both settings. For those techniques that are suitable for merely one setting, we have emphasized their limitations and stated the practical reasons behind them.

5.3.2 Horizontal FL and Vertical FL

Based on the characteristics of data distribution, FL applications can also basically be classified as horizontal FL (where clients' data have the same feature space but different samples) or vertical FL (where clients' data share the same sample space but have different features) [170]. Horizontal FL is typically initiated by a service provider who wants to improve the quality of an ML application to better fit the end users' data of a specific domain. For example, by combining the statistics of users' input habits, the developers of a mobile keyboard application could achieve a more

intelligent prediction of the next words typed by the users, thus enhancing their experience.

On the other hand, vertical FL participants are usually a few organizations who hold data in different domains while being likely to achieve win-win cooperation by knowledge sharing. For instance, a regional e-commerce company and a bank may share the same set of clients. By incorporating the knowledge of the clients' revenue and expenditure recorded at the bank as well as the purchasing traces collected by the e-commerce, they can build a more accurate prediction model on the clients' purchasing behaviors, benefiting both of their businesses.

Given that there exist diverse training workflows of vertical FL and the community has not yet achieved a consensus on the use of any one [171], [172], [173], [174], much discussion in this survey is biased towards horizontal FL that has a widely acknowledged architecture (§2.1). Still, we can provide some insights for optimizing vertical FL training in general:

1) As the data is feature-partitioned, vertical FL needs the participation of all clients in each model update attempt. Thus, optimizations on participant selection (§3.1) are not applicable to vertical FL.

2) In terms of communication, model-agnostic compression techniques (§3.2.1) should still apply to plaintext variants of vertical FL such as [175]. However, synchronization frequency reduction (§3.2.2) is not possible in vertical FL, as each client does not have a complete model and cannot conduct training independently.

3) As for local training, as all participants need processing data of the same sample space, load balancing techniques that assign a different amount of data samples to each client (§3.2.3) are not feasible in vertical FL. Moreover, as clients inherently have different local models, adaptive optimization and bias reduction methods (§3.2.4) from horizontal FL do not generalize well to vertical FL.

4) Unlike horizontal FL, the forms of aggregation in vertical FL differ significantly by the model architecture. Thus, the mentioned optimizations (§3.3) can hardly be a generic remedy.

Besides horizontal FL and vertical FL, there recently emerges a novel collaborative learning paradigm called Federated Transfer Learning (FTL) [170], [176] which can cope with more relaxed assumptions of client data distribution using transfer learning techniques [159]. Specifically, it deals with the cases where two clients' data not only differ in sample space but also feature space. Toward this end, FTL learns a common feature representation between the two clients and minimizes the empirical errors in predicting one client's labels by leveraging the labels of other clients. As FTL's training workflow ensembles some variants of vertical FL [170], we believe that the above-mentioned insights for vertical FL also hold for FTL.

5.4 Conclusion

In this survey, we focus on system optimization in synchronous federated training and propose a natural taxonomy that categorizes existing solutions based on both the training phase and the type of utility at which they target. Apart from problem-driven attempts, we also include related cornerstone efforts including measurement studies and benchmarking suites. We expect this manuscript to be a *useful guideline* for the design and implementation of federated learning systems.

REFERENCES

[1] (2016) Regulation (eu) 2016/679 of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

[2] (1996) The health insurance portability and accountability act of 1996 (hipaa). [Online]. Available: https://www.cdc.gov/phlp/publications/topic/hipaa.html

[3] (2018) California consumer privacy act (ccpa). [Online]. Available: https://oag.ca.gov/privacy/ccpa

[4] (2021) Gdpr enforcement tracker. [Online]. Available: https://www.enforcementtracker.com/

[5] (2021) Hipaa fines listed by year. [Online]. Available: https://compliancy-group.com/hipaa-fines-directory-year/

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[7] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," in *arXiv preprint*, 2018.

[8] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," in *arXiv*, 2018.

[9] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," in *arXiv*, 2019.

[10] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," in *arXiv*, 2019.

[11] (2021) Your chats stay private while messages improves suggestions. [Online]. Available: https://support.google.com/messages/answer/9327902

[12] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluivers, R. van Dalen, C. W. Lau, L. Carlson, F. Granqvist, C. Vandevelde *et al.*, "Federated evaluation and tuning for on-device personalization: System design & applications," in *arXiv preprint*, 2021.

[13] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn *et al.*, "Ibm federated learning: an enterprise framework white paper v0. 1," in *arXiv preprint*, 2020.

[14] (2020) Utilization of fate in anti money laundering through multiple banks. [Online]. Available: https://www.fedai.org/cases/utilization-of-fate-in-anti-money-laundering-through-multiple-banks/

[15] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso *et al.*, "Privacy-preserving federated brain tumour segmentation," in *MLMI Workshop*, 2019.

[16] (2020) Triaging covid-19 patients: 20 hospitals in 20 days build ai model that predicts oxygen needs. [Online]. Available: https://blogs.nvidia.com/blog/2020/10/05/federated-learning-covid-oxygen-needs/

[17] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE ICC*, 2019.

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2022.3177222

15

[19] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *INFO-COM*, 2020.

[20] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in *HPDC*, 2020.

[21] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *OSDI*, 2021.

[22] Y. G. Kim and C.-J. Wu, "Autofl: Enabling heterogeneity-aware energy efficient federated learning," in *MICRO*, 2021.

[23] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *NeurIPS*, 2017.

[24] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *NeurIPS*, 2017.

[25] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "Signsgd with majority vote is communication efficient and fault tolerant," in *ICLR*, 2019.

[26] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized sgd and its applications to large-scale distributed optimization," in *ICML*, 2018.

[27] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *International Conference on Machine Learning*, 2019.

[28] J. Jiang, F. Fu, T. Yang, and B. Cui, "Sketchml: Accelerating distributed machine learning with data sketches," in *SIGMOD*, 2018.

[29] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora, "Communication-efficient distributed sgd with sketching," in *NeurIPS*, 2019.

[30] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "Fetchsgd: Communication-efficient federated learning with sketching," in *ICML*, 2020.

[31] R. Spring, A. Kyrillidis, V. Mohan, and A. Shrivastava, "Compressing gradient optimizers via count-sketches," in *ICML*, 2019.

[32] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *NeurIPS*, 2018.

[33] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *EMNLP*, 2017.

[34] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *ICLR*, 2018.

[35] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *NeurIPS*, 2018.

[36] S. Shi, Q. Wang, K. Zhao, Z. Tang, Y. Wang, X. Huang, and X. Chu, "A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks," in *ICDCS*, 2019.

[37] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibeons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *NSDI*, 2017.

[38] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *ECML PKDD*, 2018.

[39] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *ICDCS*, 2019.

[40] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 31, no. 10, pp. 4229–4238, 2019.

[41] C. Chen, H. Xu, W. Wang, B. Li, B. Li, L. Chen, and G. Zhang, "Communication-efficient federated learning with adaptive parameter freezing," in *ICDCS*, 2021.

[42] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L.-C. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1345–1348, 2019.

[43] H. T. Nguyen, N. C. Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: a deep reinforcement learning approach," in *WF-IoT*, 2020.

[44] C. Wang, Y. Yang, and P. Zhou, "Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity," *IEEE TPDS*, vol. 32, no. 2, 2020.

[45] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *MLSys*, 2020.

[46] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *ICLR*, 2021.

[47] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, 2020.

[48] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *ICML*, 2019.

[49] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," in *NeurIPS*, 2020.

[50] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE TPDS*, vol. 31, no. 8, pp. 1754–1766, 2020.

[51] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy, "Feddane: A federated newton-type method," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019.

[52] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *ICLR*, 2021.

[53] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *ICML*, 2020.

[54] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. Reddi, S. U. Stich, and A. T. Suresh, "Breaking the centralized barrier for cross-device federated learning," in *NeurIPS*, 2021.

[55] M. Al-Shedivat, J. Gillenwater, E. Xing, and A. Rostamizadeh, "Federated learning via posterior averaging: A new perspective and practical algorithms," in *ICLR*, 2020.

[56] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *IEEE ICC*, 2020.

[57] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *IEEE ICASSP*, 2020.

[58] W. Wu, L. He, W. Lin, and R. Mao, "Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems," *IEEE TPDS*, 2020.

[59] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 479–489, 2021.

[60] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *ATC*, 2020.

[61] Z. Jiang, W. Wang, and Y. Liu, "Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning," in *arXiv*, 2021.

[62] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," in *NeurIPS Workshop*, 2019.

[63] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "Slowmo: Improving communication-efficient distributed sgd with slow momentum," in *ICLR*, 2020.

[64] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *ICLR*, 2021.

[65] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *WWW*, 2021.

[66] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," in *NeurIPS' Workshop*, 2019.

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

[68] F. Lai, Y. Dai, X. Zhu, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning," in *arXiv preprint*, 2021.

[69] (2021) Federated ai technology enabler. [Online]. Available: https://github.com/FederatedAI/FATE

[70] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu *et al.*, "Fedml: A research library and benchmark for federated machine learning," in *NeurIPS Workshop*, 2020.

[71] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," in *arXiv preprint*, 2020.

[72] (2021) Plato: A new framework for federated learning research. [Online]. Available: https://github.com/TL-System/plato

[73] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1, 2021.

[74] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *MLSys*, 2019.

[75] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[76] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[77] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021, page 25," 2021, https://www.ramonmillan.com/documentos/bibliografia/VisualNetworkingIndexGlobalMobileDataTrafficForecastUpdate2016_Cisco.pdf.

[78] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017.

[79] C. Lao, Y. Le, K. Mahajan, Y. Chen, W. Wu, A. Akella, and M. M. Swift, "Atp: In-network aggregation for multi-tenant learning." in *NSDI*, 2021, pp. 741–761.

[80] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. Ports, and P. Richtarik, "Scaling distributed machine learning with in-network aggregation," in *NSDI*, 2021.

[81] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *USENIX Security*, 2020, pp. 1605–1622.

[82] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," in *NeurIPS Workshop*, 2019.

[83] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *NeurIPS*, 2017.

[84] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *NeurIPS*, 2020.

[85] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," in *IEEE TNNLS*, 2020.

[86] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *ICML*, 2021.

[87] D. A. E. Acar, Y. Zhao, R. Zhu, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Debiasing model updates for improving personalized federated training," in *ICML*, 2021.

[88] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," in *arXiv preprint arXiv:1910.03581*, 2019.

[89] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," 2020.

[90] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *ICML*, 2021.

[91] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning over heterogeneous devices," in *arXiv*, 2021.

[92] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, 2020.

[93] C. He, M. Annavaram, and S. Avestimehr, "Towards non-iid and invisible data with fednas: federated deep learning via neural architecture search," in *arXiv*, 2020.

[94] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: a survey," *Complex & Intelligent Systems*, vol. 7, no. 2, 2021.

[95] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE S&P*, 2019.

[96] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*, 2017.

[97] C. Song and V. Shmatikov, "Auditing data provenance in text-generation models," in *SIGKDD*, 2019.

[98] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019.

[99] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *INFOCOM*, 2019.

[100] A. D. P. Team, "Learning with privacy at scale," *Apple Machine Learning Journal*, 2017.

[101] S. Shastri, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram, "Understanding and benchmarking the impact of gdpr on database systems," in *VLDB*, 2020.

[102] S. Shastri, M. Wasserman, and V. Chidambaram, "The seven sins of personal-data processing systems under {GDPR}," in *HotCloud*, 2019.

[103] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[104] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *ICML*, 2020.

[105] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Workshop*, 2013.

[106] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[107] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[108] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li, "Communication-efficient distributed deep learning: A comprehensive survey," in *arXiv*, 2020.

[109] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," in *NeurIPS*, 2019.

[110] Y. Zhang, G. Bai, X. Li, C. Curtis, C. Chen, and R. K. Ko, "Privcoll: Practical privacy-preserving collaborative machine learning," in *ESORICS*, 2020.

[111] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE S&P*, 2017.

[112] C. Liu, S. Chakraborty, and D. Verma, "Secure model fusion for distributed learning using partial homomorphic encryption," in *PADG*, 2019.

[113] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *PKC*, 2010.

[114] ——, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

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

[116] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018.

[117] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014.

[118] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013.

[119] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. Reddi, S. Kumar, and S. Sra, "Why are adaptive methods good for attention models?" in *NeurIPS*, 2020.

[120] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data *et al.*, "A field guide to federated optimization," in *arXiv*, 2021.

[121] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *ICML*, 2014.

[122] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," in *arXiv*, 2020.

[123] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *NeurIPS*, 2020.

[124] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, J. Jiang, and C. Zhang, "Multi-center federated learning," in *arXiv*, 2021.

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2022.3177222

17

[125] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[126] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *CCS*, 2020.

[127] C.-S. Yang, J. So, C. He, S. Li, Q. Yu, and S. Avestimehr, "Lightsecagg: Rethinking secure aggregation in federated learning," in *MLSys*, 2022.

[128] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," 2020.

[129] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt*, 1999.

[130] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *AAAI*, 2019.

[131] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[132] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[133] S. Reddi, M. Zaheer, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *NeurIPS*, 2018.

[134] PKU-Chengxu, "Flash," 2021, https://github.com/PKU-Chengxu/FLASH.

[135] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[136] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, 2013.

[137] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," in *arXiv preprint*, 2017.

[138] (2021) Stack overflow data. [Online]. Available: https://www.kaggle.com/stackoverflow/stackoverflow

[139] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?" in *ACL*, 2018.

[140] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, 2019.

[142] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," in *OSDI*, 2016.

[143] K. H. Li, P. P. B. de Gusmão, D. J. Beutel, and N. D. Lane, "Secure aggregation for federated learning in flower," in *Distributed ML*, 2021.

[144] MindSpore-AI, "Mindspore," 2021, https://gitee.com/mindspore/mindspore.

[145] (2021) Tensorflow federated. [Online]. Available: https://www.tensorflow.org/federated

[146] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," in *PPML*, 2018.

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

[148] A. Das and T. Brunschwiler, "Privacy is what we care about: Experimental investigation of federated learning on edge devices," in *AIChallengeIoT*, 2019.

[149] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," in *arXiv preprint*, 2020.

[150] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," in *arXiv preprint*, 2020.

[151] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, and K. E. Tan, "Achieving security and privacy in federated learning systems: Survey, research challenges and future directions," in *arXiv preprint*, 2020.

[152] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, "Privacy preservation in federated learning: Insights from the gdpr perspective," in *arXiv preprint*, 2020.

[153] R. Gu, S. Yang, and F. Wu, "Distributed machine learning on mobile devices: A survey," in *arXiv preprint*, 2019.

[154] S. Shi, Z. Tang, X. Chu, C. Liu, W. Wang, and B. Li, "A quantitative survey of communication optimizations in distributed deep learning," *IEEE Network*, vol. 35, no. 3, pp. 230–237, 2020.

[155] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[156] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[157] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," in *arXiv*, 2021.

[158] S. Li, L. Wang, W. Wang, Y. Yu, and B. Li, "George: Learning to place long-lived containers in large clusters with operation constraints," in *ACM SoCC*, 2021.

[159] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, 2009.

[160] I. Ergun, H. U. Sami, and B. Guler, "Sparsified secure aggregation for privacy-preserving federated learning," in *arXiv*, 2021.

[161] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[162] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, 2017.

[163] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *ICML*, 2018.

[164] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.

[165] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[166] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Security & Privacy*, 2019.

[167] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 57–64.

[168] P. Jauernig, A.-R. Sadeghi, and E. Stapf, "Trusted execution environments: properties, applications, and challenges," *IEEE Security & Privacy*, vol. 18, no. 2, 2020.

[169] M. Orenbach, Y. Michalevsky, C. Fetzer, and M. Silberstein, "{CoSMIX}: A compiler-based system for secure memory instrumentation and execution in enclaves," in *ATC*, 2019.

[170] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[171] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A quasi-newton method based vertical federated learning framework for logistic regression," in *NeurIPS Workshop*, 2019.

[172] Y. Hu, D. Niu, J. Yang, and S. Zhou, "Fdml: A collaborative machine learning framework for distributed features," in *KDD*, 2019.

[173] B. Gu, Z. Dang, X. Li, and H. Huang, "Federated doubly stochastic kernel learning for vertically partitioned data," in *KDD*, 2020.

[174] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, 2021.

[175] S. Feng and H. Yu, "Multi-participant multi-class vertical federated learning," in *arXiv*, 2020.

[176] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, 2020.

Zhifeng Jiang received the BEng (Hons.) degree from the Department of Computer Science and Technology at Zhejiang University (ZJU) in 2019. Since then, he has been working towards a Ph.D. degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST). He is mainly interested in machine learning systems, with a special focus on efficient and scalable federated learning. He was a recipient of the Best Paper Runner-up Award at IEEE ICDCS 2021. He served on the artifact evaluation committees of ACM SOSP 2021, USENIX OSDI 2022, and USENIX ATC 2022.
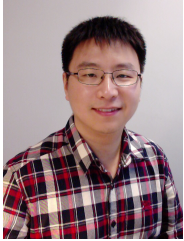
Wei Wang (Member, IEEE) received his B.Engr. and M.Engr. degrees from the Department of Electrical Engineering, Shanghai Jiao Tong University, China, in 2007 and 2010, respectively and his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2015. Since 2015, he has been with the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST), where he is currently an Associate Professor. He is also affiliated with the HKUST Big Data Institute. Dr. Wang's research interests cover the broad area of distributed systems, with focus on serverless computing, machine learning systems, and cloud resource management. He published extensively in the premier conferences and journals of his fields. His research has won the Best Paper Runner Up awards of IEEE ICDCS 2021 and USENIX ICAC 2013.

Bo Li is a Chair Professor in Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He was a Cheung Kong Scholar Visiting Chair Professor in Shanghai Jiao Tong University (2010-2016), and was the Chief Technical Advisor for ChinaCache Corp. (NASDAQ:CCIH), a leading CDN provider. He made pioneering contributions in multimedia communications and the Internet video broadcast, which attracted significant investment and received the Test-of-Time Best Paper Award from IEEE INFOCOM (2015). He received 6 Best Paper Awards from IEEE including IEEE INFOCOM (2021). He was the Co-TPC Chair for IEEE INFOCOM (2004).

He is a Fellow of IEEE. He received his PhD in the ECE Department, University of Massachusetts at Amherst, and his B. Eng. (summa cum laude) in the Computer Science from Tsinghua University, Beijing, China.

Qiang Yang received the B.Sc. degree in astrophysics from Peking University, Beijing, China, in 1982, and the Ph.D. degree in computer science and the M.Sc. degree in astrophysics from the University of Maryland at College Park, College Park, MD, USA, in 1985 and 1989, respectively.

He was a Faculty Member with the University of Waterloo, Waterloo, ON, Canada, from 1989 to 1995, and Simon Fraser University, Burnaby, BC, Canada, from 1995 to 2001. He was the Founding Director of the Noah's Ark Laboratory, Huawei, Hong Kong, from 2012 to 2014, and a Co-Founder of 4Paradigm Corporation, Beijing, an AI platform company. He is currently the Head (Chief AI Officer) with the AI Department, WeBank, Shenzhen, China, and the Chair Professor with the Department of Computer Science and Engineering (CSE), The Hong Kong University of Science and Technology, Hong Kong, where he was the Former Head of the Department of CSE and the Founding Director of the Big Data Institute, Hong Kong, from 2015 to 2018. He has authored several books, including Intelligent Planning (Springer), Crafting Your Research Future (Morgan and Claypool), and Constraint-Based Design Recovery for Software Engineering (Springer). His research interests include artificial intelligence, machine learning, and data mining, with an emphasis on transfer learning, automated planning, federated learning, and case-based reasoning.

Dr. Yang is a fellow of several international societies, including the ACM, AAAI, IAPR, and AAAS. He served as an Executive Council Member for the Association for the Advancement of AI from 2016 to 2020 and the President for the International Joint Conference on AI from 2017 to 2019. He was a recipient of several awards, including the 2004/2005 ACM KDDCUP Championship, the AAAI Innovative AI Applications Award in 2016, and the ACM SIGKDD Distinguished Service Award in 2017. He was the Founding Editor-in-Chief of the ACM Transactions on Intelligent Systems and Technology and the IEEE TRANSACTIONS ON BIG DATA.