FLASH: Heterogeneity-Aware Federated Learning at Scale

Chengxu Yang, Mengwei Xu *Member, IEEE*, Qipeng Wang, Zhenpeng Chen, Kang Huang, Yun Ma *Member, IEEE*, Kaigui Bian *Senior Member, IEEE*, Gang Huang *Member, IEEE*, Yunxin Liu *Member, IEEE*, Xin Jin *Member, IEEE*, and Xuanzhe Liu *Senior Member, IEEE*

Abstract—Federated learning (FL) becomes a promising machine learning paradigm. The impact of heterogeneous hardware specifications and dynamic states on the FL process has not yet been studied systematically. This paper presents the first large-scale study of this impact based on real-world data collected from 136k smartphones. We conducted extensive experiments on our proposed heterogeneity-aware FL platform namely `FLASH`, to systematically explore the performance of state-of-the-art FL algorithms and key FL configurations in heterogeneity-aware and -unaware settings, finding the following. (1) Heterogeneity causes accuracy to drop by up to 9.2% and convergence time to increase by 2.32×. (2) Heterogeneity negatively impacts popular aggregation algorithms, e.g., the accuracy variance reduction brought by *q-FedAvg* drops by 17.5%. (3) Heterogeneity does not worsen the accuracy loss caused by gradient-compression algorithms significantly, but it compromises the convergence time by up to 2.5×. (4) Heterogeneity hinders client-selection algorithms from selecting wanted clients, thus reducing effectiveness. e.g., the accuracy increase brought by the state-of-the-art client-selection algorithm drops by 73.9%. (5) Heterogeneity causes the optimal FL hyper-parameters to drift significantly. More specifically, the heterogeneity-unaware setting favors looser deadline and higher reporting fraction to achieve better training performance. (6) Heterogeneity results in non-trivial failed clients (more than 10%) and leads to participation bias (the top 30% of clients contribute 86% of computations). Our FLASH platform and data have been publicly open sourced.

Index Terms—Federated learning, heterogeneity, impact analysis

1 Introduction

In recent years, machine learning (ML) has undergone a paradigm shift from cloud data centers toward mobile devices [1], [2], [3], [4]. Mobile devices routinely collect user behavior/usage data and these real-world data can be used to train high-quality ML models. However, with increasing concerns about user privacy, many policies and legislation formulations are enforced to regulate the use of private data e.g., the GDPR [5] and CCPA [6]. As a result, the emerging *federated learning* (FL) algorithm [7] has drawn tremendous attention due to its potential privacy-preserving advantages [2], [8]. The key idea of FL is to use a set of personal mobile devices to train an ML model collaboratively under the orchestration of a central server. Since the FL training process takes place on mobile devices without uploading user data beyond them, it is regarded to be quite promising for preserving user privacy. Therefore, FL has been widely adopted by industry leaders such as Google, Apple, and NVIDIA for various ML tasks including query suggestion [9], natural language processing [10], medical imaging AI [11], etc.

Essentially, FL relies on numerous and distributed mobile devices to train and share ML models. These devices (e.g., personal smartphones) have certain inherent characteristics that set FL apart from traditional distributed ML. These characteristics are conceptually called *heterogeneity* [8]. In particular, heterogeneity can be attributed to two major aspects. One is from hardware specifications of devices (called *hardware heterogeneity*), e.g., different capabilities of CPU, RAM, and battery life. Additionally, the state and running environment of participating devices may be various and dynamic (called *state heterogeneity*), e.g., CPU busy/free, stable/unreliable network connections to the server, etc.

Intuitively, heterogeneity can impact FL in terms of accuracy and training time. For instance, it is not surprising when a device fails to upload its local model updates to the server, and this can affect the training time to obtain a converged global model. Furthermore, devices that seldom participate in an FL task due to unexpected states (e.g., frequently interrupted by users) can be under-represented by the global model.

To the best of our knowledge, most FL researchers take a simulation approach to evaluate their algorithms, given the high cost of field deployment [7], [12], [13], [14], [15], [16]. In this case, they usually ignore heterogeneity and make an overly idealistic assumption, i.e., all devices are always available for training and equipped with *homogeneous* hard-

- *Chengxu Yang, Qipeng Wang, Kaigui Bian, Gang Huang, Xin Jin, and Xuanzhe Liu are with School of Computer Science at Peking University, Beijing, China. E-mail: {yangchengxu, bkg, hg, xinjinpku, xzl}@pku.edu.cn, wangqipeng@stu.pku.edu.cn*
- *Mengwei Xu is with the Computer Science Department at Beijing University of Posts and Telecommunications, Beijing, China. E-mail: mwx@bupt.edu.cn*
- *Zhenpeng Chen is with the Department of Computer Science, University College London, London, United Kingdom. E-mail: zp.chen@ucl.ac.uk*
- *Kang Huang is with the Linggui Tech, Beijing, China. E-mail: kang.huang@nlptech.com*
- *Yun Ma is with the Institute for Artificial Intelligence, Peking University, Beijing, China. E-mail: mayun@pku.edu.cn*
- *Yunxin Liu is with the Institute for AI Industry Research (AIR) at Tsinghua University, Beijing, China. E-mail: liuyunxin@air.tsinghua.edu.cn*

ware specifications (e.g., the same CPU and RAM capacity). What is more, since most research efforts have not accessed real-world large-scale FL systems, they usually apply an ideal set of system configurations. For example, these efforts subconsciously set the reporting deadline (deadline for devices to finish the job in a round) to a very loose value (e.g., unlimited) and expect that all devices can upload successfully and promptly. Although some recent studies [17], [18], [19], [20], [21] have realized the heterogeneity in FL, they still fail to evaluate in a heterogeneity-aware environment due to the lack of real-world data describing heterogeneity among devices [1]. In summary, the impact of heterogeneity has not yet been systematically demonstrated and measured.

In this work, we carry out the first systematic study to measure the impact of heterogeneity on FL. To this end, we have developed a holistic platform that complies with the standard and widely adopted FL protocol [1], [9], [10] but enables us to evaluate existing FL algorithms in a *heterogeneity-aware* environment (i.e., considering devices with dynamic states and various hardware capacities) for the first time. The platform is powered by a large-scale real-world dataset that faithfully reflects heterogeneity across numerous devices. To build this dataset, we collected the device hardware specifications and regular state changes (including the states related to device check-in and drop-out in FL) of 136k smartphones in one week through a commodity input method app (IMA).

Based on the data and platform, we conducted extensive experiments (>7,000 GPU-hours on the cloud) to measure the impact of heterogeneity on FL by comparing the results of different FL algorithms and configurations in heterogeneity-aware and heterogeneity-unaware settings. We selected four representative FL tasks for study, including two image-classification tasks and two natural language processing tasks. For every task, we employed a benchmark dataset for model training. Three of the benchmark datasets [22], [23], [24] have been widely used in existing FL-related studies [7], [12], [13], [17], [25], and the last one is a real-world text input dataset collected from the aforementioned IMA.

In summary, we make the following contributions:

• **Dataset.** We establish a large-scale dataset that describes heterogeneity among 136,000 real-world mobile devices. To the best of our knowledge, it is the first dataset of this kind and on this scale, and can easily be incorporated into existing FL platforms.

• **Platform.** We built an FL platform, namely `FLASH`, which is the first known heterogeneity-aware environment for evaluating the impact raised by heterogeneity in popular FL algorithms and configurations.

• **Findings and implications.** We conducted extensive measurement experiments to quantify the impact of heterogeneity and investigate the influential factors behind this impact. Based on the results, we derive the following findings and implications.

- Heterogeneity has a non-trivial impact on the performance of FL algorithms. For basic FL algorithms (e.g., *FedAvg*), heterogeneity causes up to a 9.2% accuracy drop and a 2.64× longer training time; heterogeneity also has a significant impact on the effectiveness of advanced FL optimization algorithms (e.g., aggregation algorithms, gradient-compression algorithms, and client-selection algorithms). For example, heterogeneity causes the accuracy variance reduction brought by *q-FedAvg*, a state-of-the-art aggregation algorithm pursuing fair contributions of participating devices in FL, to drop by 17.5%; heterogeneity lengthens the training time of gradient-compression algorithms by up to 2.5×; and it hinders FL client-selection algorithms from selecting wanted clients, thus making them less effective. These findings urge FL algorithm designers to consider necessary heterogeneity.
- Heterogeneity indeed impacts FL configurations. Specifically, the optimal settings on deadline and reporting fraction, two key system configurations in FL, are significantly different in heterogeneity-aware and heterogeneity-unaware settings. In heterogeneity-unaware settings, one could simply choose a loose enough deadline and set the reporting fraction to 1.0; in heterogeneity-aware settings, one needs to adjust them carefully according to the proportion of failed clients. These findings suggest that heterogeneity-aware FL platforms are needed for FL developers to adjust the configurations.
- We dive deeper into the results and find that state heterogeneity has more influence over the accuracy drop than hardware heterogeneity (9.5% vs. 0.4%). Moreover, we identify two influential factors that account for the aforementioned impact of heterogeneity. (1) *Device failure*: on average, 11.6% of selected devices fail to upload their model updates per round due to an unreliable network connection, excessive training time, and drop-out caused by user interruption. This failure slows down model convergence and wastes valuable hardware resources. (2) *Participant bias*: devices go through the FL process in a biased manner. For instance, we find that more than 30% of devices never participate in the learning process when the model converges and the global model is dominated by active devices. These findings suggest that a "proactive alerting" technique that reduces device failure and a selection scheme that resolves bias are promising to mitigate the impact of heterogeneity.

• **Open source.** We make the platform, scripts, and data used in this study available [2] as an additional contribution to the research community for other researchers to reproduce, replicate, extend, and build upon.

Some of the results in this paper were reported in our previous work [26]. We expand on the conference version in the following aspects. (1) We have extended our dataset and improved our platform to better incorporate heterogeneity (see Sections 3.2.1 and 3.3). (2) We have extended our research scope to systematically investigate the impact of heterogeneity on client-selection optimization (see Section 5.3) and FL configurations (see Section 6). (3) We have extended Section 2 to provide comprehensive background knowledge and compare state-of-the-art FL research.

The rest of this paper is organized as follows. Section 2 describes the background of this work and related research,

[1]Large companies have built practical FL systems [1], but detailed data is not disclosed.

[2]https://github.com/PKU-Chengxu/FLASH

and proposes our research questions. Section 3 describes the methodology of this measurement study, including the collected dataset, proposed FL platform, and detailed experimental settings. Sections 4 to 7 answer the research questions based on the evaluation results. Section 8 summarizes practical implications derived from the findings for FL stakeholders. Section 9 discusses the threats that could affect the validity of this study, and is followed by concluding remarks in Section 10.

2 Background and Motivation

In this section, we first introduce the background knowledge and work related to this study. Then we propose our research questions and describe what motivated them.

2.1 Preliminaries and Related Work

FL is an emerging privacy-preserving learning paradigm. In this paper, we focus on the most widely studied scenario of FL – *cross-device* FL [7], [27] – which utilizes a federation of *client devices*[3], coordinated by a *central server*, to train a global ML model. Its typical workflow [1] consists of several rounds, where each round can be divided into three phases: (1) the central server first selects devices to participate in the FL; (2) each selected device retrieves the latest global model from the server as the local model, re-trains the local model with local data, and uploads the updated weights/gradients of the local model to the server; (3) the server finally aggregates the updates from devices and obtains a new global model.

In practice, FL is typically implemented based on state-of-the-art FL algorithms, such as *FedAvg* [7]. *FedAvg* is a representative FL algorithm that has been widely used in the FL literature [13], [15], [16], [28] and deployed in the industry, e.g., in Google's production FL system [20]. In *FedAvg*, devices perform multiple local training epochs, where a device updates the weights of its local model using its local data each round. Then the central server averages the updated weights of local models as the new weights of the global model.

Furthermore, many advanced algorithms have been proposed to optimize FL, including reducing the communication cost between the central server and devices [12], [29], [30], [31], [32], [33], [34], accelerating the training process [17], [21], [35], enhancing the privacy guarantee or attacking FL models [14], [36], [37], [38], [39], [40], [41], [42], ensuring fairness across devices [13], [15], [16], minimizing the on-device energy cost [43], [44], [45], incentivizing data owners [46], etc. Notably, most of them have not been well evaluated in a heterogeneity-aware environment, making their benefits unclear for real-world deployment.

Apart from the FL algorithms, additional system configurations are required to implement FL in real-world applications [1], [9]. Here, we describe some key configurations. (1) *Reporting deadline* (*deadline* for short) specifies how long the server waits for the selected clients to finish their job in each round, including transmission and model training. (2) *Reporting fraction* specifies how many clients are needed to commit a round. In each round, if the proportion of successfully uploaded clients is smaller than the fraction, this round will be discarded and the global model will not be updated (fail to commit this round). (3) *Goal client count* specifies how many clients are selected to participate in the training each round. (4) *Selection time window* specifies how long the server waits when it is selecting clients. Typically, a round will be abandoned and restarted if the number of available clients is less than the *goal client count*. (5) *Maximum sample count* is the maximum number of samples used for training on every single device, which is set to balance the training time on devices. Although previous FL systems have reported their configurations [1], [9] on their tasks, most FL researchers with no access to FL systems still question how to choose proper configurations since they are usually task-specific. As a result, most FL research [13], [14], [15], [16] has to simplify its assumptions and set configurations to an ideal value. For example, the deadline is usually assumed to be unlimited because each selected device is expected to finish its job successfully and promptly [7]. Obviously, that will not be the case once heterogeneity is considered. However, the impact of heterogeneity on FL configurations has not been studied systematically.

Heterogeneity is one of the characteristics that set FL apart from traditional distributed ML because the training process in FL usually happens on numerous mobile devices. Following existing work [2], [8], we define heterogeneity as *(1) the various hardware specifications of participating devices (called hardware heterogeneity, e.g., different CPU, RAM, and battery life) and (2) the dynamic and changeable states and running environments of participating devices (called state heterogeneity, e.g., CPU busy/free, stable/unreliable network connections to the server, etc.)*.

Intuitively, heterogeneity could affect FL because mobile devices have various hardware capacities (resulting in various training speeds) and their states are changeable depending on the devices' owners. To the best of our knowledge, most existing FL research ignores heterogeneity [7], [12], [13], [14], [15] – a typical assumption is that all devices are always available for training and have consistent training performance. A small number of works recognize the existence of heterogeneity in FL and take some steps to handle it [17], [18], [19], [20]. Unfortunately, it is still questionable how these methods would perform in real FL systems. For example, *FedProx* [17] handles hardware heterogeneity by allowing each participating device to perform a variable amount of work, but the hardware capability of each device is randomly set and changes in device state remain unconsidered.

2.2 Research Questions

An important and natural characteristic of FL – heterogeneity and its impact – has not been well investigated in the FL literature, which motivated us to perform this study. Specifically, we aim to answer the following research questions (RQs).

• **RQ1 (Impact on basic FL algorithms):** *How does heterogeneity affect basic FL algorithms?* FedAvg [7], as the state-of-the-art algorithm to implement FL, has been widely adopted in the industry [1], [9], [10], [47] and frequently used as a baseline in the FL literature [13], [16], [17]. Considering its importance for FL, we use it as a representative basic FL

[3]In the rest of this paper, we use *device* to refer to *client device*.

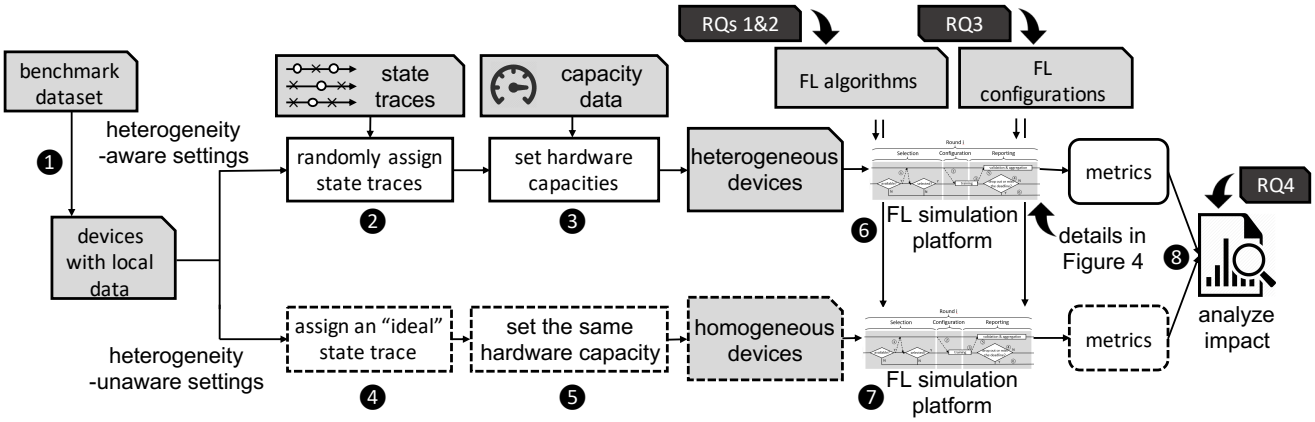


Fig. 1. Overview of our methodology.

algorithm and aim to measure whether and how it would be affected by heterogeneity.

• **RQ2 (Impact on advanced FL algorithms):** *How does heterogeneity affect advanced FL algorithms?* Optimizations [7], [12], [13], [17], [43], [48], i.e., advanced algorithms, are proposed on top of FL. Those techniques, however, have not been evaluated in a heterogeneity-aware environment. Hence, we study how heterogeneity affects the effectiveness of these techniques. Specifically, we select three important and representative categories of algorithms: aggregation algorithms (a fundamental component in FL), gradient-compression algorithms (important because massive client-server communication overheads are one of the challenges in FL [2]), and client-selection algorithms (an emerging and promising trend in FL that is effective in improving model accuracy). These three types of algorithms correspond to the three phases in FL (see Section 3.3). The results shed light on the criteria or factors in designing corresponding algorithms. Specifically, we dive deeper by answering three specific questions.

- **RQ 2.1 (Impact on aggregation algorithms):** *How does heterogeneity affect aggregation algorithms?* The aggregation algorithm is a key component in FL that determines how to aggregate the model updates from clients. By evaluating them under heterogeneity-aware and heterogeneity-unaware settings, we investigate whether and to what extent their performance is affected.
- **RQ 2.2 (Impact on gradient-compression algorithms):** *How does heterogeneity affect gradient-compression algorithms?* The gradient-compression algorithm is usually applied to reduce the communication cost between the server and clients. Considering that gradient compression is usually achieved at the expense of accuracy, we investigate whether heterogeneity would exacerbate this effect.
- **RQ 2.3 (Impact on client-selection algorithms):** *How does heterogeneity affect client-selection algorithms?* The client-selection algorithm determines the selected clients in each round of the FL workflow. It is inherently related to heterogeneity because the available clients frequently change over time under heterogeneity-aware settings.

• **RQ3 (Impact on FL configurations):** *How does heterogeneity affect FL configurations?* As presented in Section 2.1, there are many key system configurations in FL. However, how to choose proper configurations according to FL tasks is still an open question. Our study aims to search for and compare optimal FL configurations in different settings (i.e., heterogeneity-aware and heterogeneity-unaware), to provide actionable implications for FL configuration issues.

• **RQ4 (Factors that influence impact):** *What are the factors that influence heterogeneity?* Besides identifying and measuring impact, we are also interested in the reasons behind it. Revealing the factors influencing heterogeneity will shed light on how to relieve its impact in future FL research.

3 The Measurement Methodology

In this section, we introduce the approach and the experimental settings of our study.

3.1 Approach Overview

Figure 1 illustrates the overall workflow of our measurement approach. It starts from a *benchmark dataset* that is typically partitioned into thousands or millions of devices holding local data for training (❶). For a fair comparison, we always used the same partition strategy in the heterogeneity-aware and heterogeneity-unaware settings, i.e., the local training data on a given device were the same.

For heterogeneity-aware settings, we randomly assigned a state trace (❷) and a hardware capacity (❸) to each device. A state trace determines whether a device is available for local training at any simulation timestamp, while the hardware capacity specifies the training speed and communication bandwidth. Both datasets were collected from large-scale real-world mobile devices through an IMA app (details in Section 3.2). As a result, we got a heterogeneous device set with different local training data, hardware capacities, and state change dynamics.

For heterogeneity-unaware settings, we assigned each device an "ideal" state trace, i.e., the device always stayed available for local training and never dropped out (❹), and a uniform hardware capacity as the mid-end device in our IMA dataset (Redmi Note 8) (❺). As a result, we got a homogeneous device set with the same hardware capacity and state change dynamics, as existing FL platforms do.

We next deployed the two device sets to our FL simulation platform and executed the FL task (e.g., image classification) under the same configurations and using the same FL algorithms (❻ and ❼). The simulation platform extends the standard FL protocol considering heterogeneity, e.g., a device can quit training due to a state change (details in Section 3.3). We finally analyzed heterogeneity's impact by comparing the metric values achieved by heterogeneous devices and homogeneous devices (❽).

3.2 The Datasets

As described in Section 3.1, we used two types of datasets in this study, including (1) the *IMA dataset* describing heterogeneity in real-world smartphone usage, and (2) benchmark datasets containing devices' local data used for training and testing ML models.

3.2.1 *IMA dataset*

To power the heterogeneity-aware settings, we collected large-scale real-world data from a popular IMA that can be downloaded from Google Play. The dataset can be divided into two parts, including (1) *device state traces* for annotating state heterogeneity, and (2) *capacity data* for annotating hardware heterogeneity.

• **Device state traces** record the state changes (including battery charging, battery level, network environment, screen locking, etc.) of 136k devices over one week starting from Jan. 31, 2020. More specifically, every time the aforementioned state changed, the IMA recorded it with a timestamp and saved it as a state entry (refer to Table 1).

In total, we collected 136k traces (one for each device) containing 180 million state entries, accounting for 111GB of storage.

The state traces determine the time intervals when a device is available for local training, which are critical in understanding FL performance in heterogeneity-aware settings. Figure 2 concretely exemplifies how a trace works during the simulation. The device becomes available for training at T_2 because it meets the state criteria, i.e., when a device is idle, charged, and connected to WiFi (these criteria are set by practical FL systems to protect the user experience [1]). Then after a period of time at T_3, the network environment changes to "4G", thus the device becomes unavailable. As a result, we obtain a training-available interval between T_2 and T_3. The device may begin to participate in FL at any time in an available interval, so it can drop out because it does not meet the criteria (as shown in the upper part of Figure 2). Similarly, if the device finishes its job before its state changes, it will succeed in uploading in this round (as shown in the lower part of Figure 2).

As far as we know, this is the first-of-its-kind device-usage dataset collected from large-scale real-world devices, making it much more representative than datasets covering a small group of devices [49], [50].

• **Hardware capacity data** indicate the computational and communication capacities of different devices. This dataset, along with the aforementioned state trace, determines how long a device can undergo local training for and whether it can complete the local training and upload the model updates to the central server before a deadline.

Field	Description	Example
user_id	Anonymized user id.	xxxyyyzzz
device_model	device type	SM-A300M
screen_trace	screen on or off	screen_on
screen_lock_trace	screen lock or unlock	screen_lock
time	time in current state	2020-01-29 05:52:16
network_trace	network condition	2G/3G/4G/5G/WiFi
battery_trace	battery charging state, battery level	battery_charged_off 96.0%

TABLE 1
Example of a state entry.

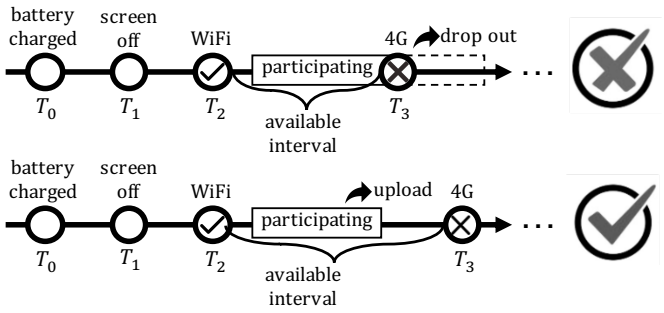


Fig. 2. A trace is a series of state changes over time. In each round, a client with a trace may drop out (the top figure) or succeed in uploading (the bottom figure), depending on the device states at the time.

For the computational capacity, we sought to obtain the training speed of a given device on a given ML model. Considering the massive number of device types (more than one thousand types, according to our collected IMA dataset), it would be too costly to profile all of them. Instead, inspired by previous work [51], we employed a "regression" approach, which is illustrated in Figure 3.

The key idea is to train a simple but effective regression model that takes a device's hardware performance data (e.g., CPU performance and memory) as the input and predicts the device's training performance (i.e., training speed on the given ML model). To train this model, we first selected 20 devices that were representative of our IMA dataset. These devices ranged from low-end to high-end devices. We profiled and performed on-device training on these representative devices and used these ground-truth data to train a regression model. We implemented on-device

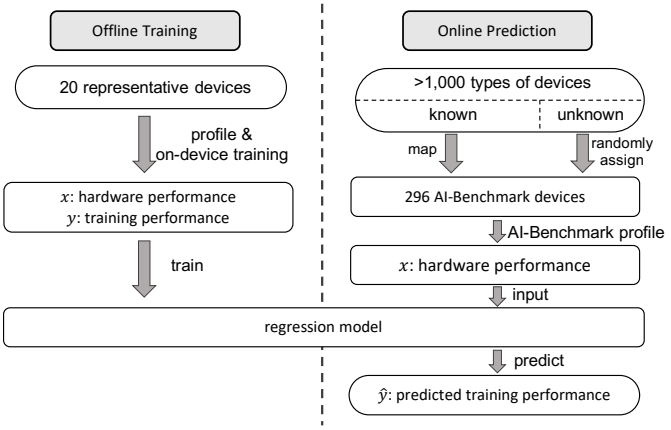


Fig. 3. The workflow of predicting the training performance of a device.

training using a state-of-the-art ML library (MNN [52]) and recorded the training time for each ML model used in our experiments as the training performance. Then given an arbitrary type of device, we leveraged profiling results from previous work [53] as the input of our regression model. The profiling results come from *AI-Benchmark*, a comprehensive AI performance benchmark that provides detailed data on various (up to 296, at the time we carried out our study) devices' hardware performance (e.g., CPU and memory score, INT8/FP16 speed, etc.). If the given device had been profiled by AI-Benchmark (noted as "known"), we directly mapped them to the corresponding device. Otherwise (noted as "unknown"), we randomly assigned an AI-Benchmark device to it. Finally, we input the AI-Benchmark profiled data into the regression model and got the predicted training performance. What is more, considering that a new ML model that we have not profiled could be provided by FL developers, we also provide a kernel-level prediction, following previous work [51].

For the communication capacity, we recruited 30 volunteers and deployed a testing app on their devices to periodically (i.e., every two hours) obtain the downstream/upstream bandwidth between the devices and a cloud server. We fitted each volunteer's data to a normal distribution and randomly assigned a distribution to the device during the simulation. The bandwidth data determined the model uploading/downloading time during the simulation.

3.2.2 *Benchmark datasets*

We used four benchmark datasets to quantitatively study the impact of heterogeneity on FL performance. Three of them (i.e., Reddit [22], Femnist [24], and Celeba [23]) are synthetic datasets widely adopted in the FL literature [8], [13], [14], [18], while the other is a real-world input corpus collected from our IMA, named M-Type. M-Type contains text input from the devices covered in the state traces in Section 3.2.1.[4] Each dataset can be used for an FL task. Specifically, Femnist and Celeba are for image-classification tasks, while Reddit and M-Type are for next-word-prediction tasks. For Femnist and Celeba, we used CNN models, and for Reddit and M-Type, we used LSTM models. The four models were implemented by *Leaf* [25], a popular FL benchmark. All the datasets are non-IID, i.e., the data distribution is skewed and unbalanced across devices, which is a common data distribution in FL scenarios [2]. We randomly split the data on each device into training/testing sets (80%/20%).

3.2.3 *Ethical considerations*

All the data were collected with the explicit agreement of users through user-term statements and a strict policy regarding data collection, transmission, and storage. The IMA users were given an explicit option to opt out of having their data collected. In addition, we took very careful steps to protect user privacy and preserve the ethics of our research. First, our work was approved by the Research Ethics Committee of the institutes that the authors are currently affiliated with. Second, the users' identities were all completely anonymized during the study. Third, the data were stored and processed on a private, HIPPA-compliant cloud server, with strict access authorized by the company that developed the IMA. The whole process was compliant with the company's privacy policy.

[4]Due to privacy concerns, we do not include M-Type in our GitHub repository.

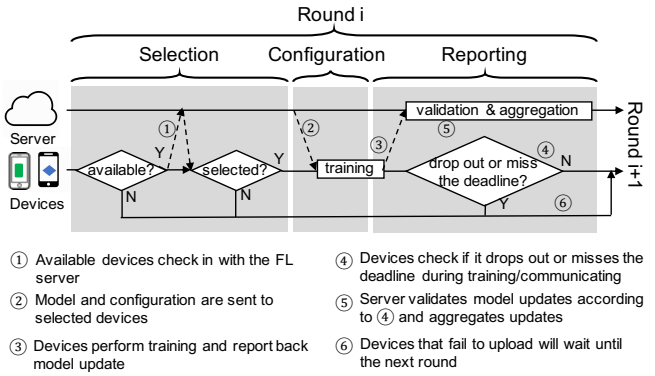


Fig. 4. The FLASH simulation platform on top of standard FL protocols [1].

3.3 The FLASH Simulation Platform

We built a simulation platform named FLASH, which follows the standard FL protocol [1] and divides the simulation into three main phases, as shown in Figure 4. We also followed Google's report [9] to configure FLASH, e.g., the time that the server waits for devices to check in. Given an FL task, a global ML model is trained in a synchronized way and advanced round by round.

Selection. At the beginning of each round, the server waits for tens of seconds for devices to check in. Devices that meet the required state criteria check in to the server (①). Then the server randomly selects a subset (by default 100) of these training-available devices.

Configuration. The server sends the global model and configuration to each of the selected devices (②), which instructs the device to train the model. The device starts to train the model using its local data once the transmission is complete (③).

Reporting. The server waits for the participating devices to report updates. The time that the server waits is configured by the *reporting deadline*. Each device first checks its "reporting qualification" (④), i.e., whether it has dropped out according to its states over the corresponding period. It also checks if it has missed the deadline according to the time needed to finish training and communication. The preceding check is powered by our IMA dataset, described in Section 3.2.1. The server validates updates based on the results of the checks and aggregates the qualified updates (⑤). Devices that fail to report and those that are not selected will wait until the next round (⑥). This reporting qualification step is what enables heterogeneity-aware FL and distinguishes our platform from existing ones.

Additional system configurations. As described above, our platform supports some additional system configurations, which are powered by the aforementioned IMA dataset. These configurations are important and necessary when deploying FL in the real world and, to the best of our knowledge, our platform is the first FL platform to support these configurations. The additional configurations include

Algorithms	Acc.	Training Time/Round	Compression Ratio	Var. of Acc.
FedAvg	✓	✓	–	–
Structured Updates	✓	✓	✓	–
GDrop	✓	✓	✓	–
SignSGD	✓	✓	✓	–
q-FedAvg	✓	✓	–	✓
FedProx	✓	✓	–	–
Oort	✓	✓	–	–
FedCS	✓	✓	–	–

TABLE 2
Three categories of FL algorithms we chose and the corresponding metrics we measured.

reporting deadline, *reporting fraction*, *goal client count*, *selection time window*, and *maximum sample count* (please refer to Section 2 for more detail about these configurations). Among them, *reporting deadline* and *reporting fraction* are two key configurations that directly control whether a round can be successfully committed. We will investigate the impact of heterogeneity on their selection and provide empirical guidance on how to properly configure them in Section 6.

3.4 Experimental Settings

Algorithms. We briefly introduce the algorithms explored in our study and provide more details and their hyper-parameters in the following sections. The algorithms can be divided into four categories: (1) the basic algorithm, i.e., *FedAvg*, which has been deployed to real systems [1] and is widely used in the FL literature [12], [13], [15], [16]; (2) aggregation algorithms that determine how to aggregate the weights/gradients uploaded from multiple devices, including *q-FedAvg* [13] and *FedProx* [17]; (3) compression algorithms, including *Structured Updates* [12], *Gradient Dropping* (*GDrop*) [54], and *SignSGD* [55], which compress local models' weights/gradients to reduce the communication cost between devices and the central server; and (4) client-selection algorithms, including *Oort* [21] and *FedCS* [18], which determine the selected clients in each round to improve model accuracy.

Metrics. In our experiments, we quantify the impact of heterogeneity by reporting the following metrics: (1) *convergence accuracy*, which is directly related to the performance of an algorithm; (2) *training time/round*, which is defined as the time/rounds for the global model to converge (we note that the training time reported by our simulation platform is the running time after the FL system is deployed in the real world, instead of the time to run the simulation purely on the cloud); (3) *compression ratio*, which is defined as the fraction of the size of the compressed gradients to the original size [56]; (4) *variance of accuracy*, which is calculated as the standard deviation of accuracy across all the devices in the benchmark dataset and indicates the cross-device fairness of an algorithm. Table 2 summarizes the algorithms and the corresponding metrics that we measure.

Computing Environment. All experiments were performed on a high-performance computing cluster with Red Hat Enterprise Linux Server release 7.3 (Maipo). The cluster had 10 GPU workers. Each worker was equipped with two Intel Xeon E5-2643 V4 processors, 256G of main memory, and 2 NVIDIA Tesla P100 graphics cards. In total, the reported experiments cost more than 7,000 GPU-hours.

4 RQ1: Impact on Basic Algorithms

We first measured the impact of heterogeneity on the performance (in terms of accuracy and training time/rounds) of the basic *FedAvg* algorithm. To obtain a more reliable result, we performed the measurement under different numbers of local training epochs, i.e., different numbers of times that the devices used their local data to update the weights of their local models (see Section 2). The number of local training epochs is an important *FedAvg* hyper-parameter used to balance the communication cost between the server and devices [7], [16], [17]. We followed previous work [7] in setting this number (denoted as E) as 1, 5, and 20. We also used the learning rate and batch size recommended by *Leaf* [25] for each ML model. Figure 5 illustrates how accuracy changes with training time and training rounds under different numbers of local training epochs. We summarize our observations and insights as follows.

Dataset	Heter.	Algo.	Average	Worst 10%	Best 10%	Var. $\times 10^{-4}$
Femnist	Unaware	*FedAvg*	82.13%	61.1%	**97.2%**	213
		q-FedAvg	**82.66%**	**64.7%**	95.1%	**157** (26.3% ↓)
	Aware	*FedAvg*	81.22%	61.1%	94.9%	203
		q-FedAvg	**81.24%**	**64.7%**	**95.1%**	**159** (21.7% ↓)
M-Type	Unaware	*FedAvg*	**8.15%**	2.33%	**13.5%**	19
		q-FedAvg	7.78%	**2.33%**	13.0%	**17** (10.5% ↓)
	Aware	*FedAvg*	7.47%	2.27%	12.3%	16.2
		q-FedAvg	**7.47%**	**2.33%**	**12.4%**	**15.6** (3.7% ↓)

TABLE 3
Test accuracy for *q-FedAvg* and *FedAvg*. "Var" represents the variance in accuracy across devices.

• **Heterogeneity causes a non-trivial accuracy drop in FL.** In heterogeneity-aware settings, accuracy decreases on each dataset across various local training epochs, specifically by an average of 2.3%, 0.5%, and 4% on the existing Femnist, Celeba, and Reddit datasets, respectively. The accuracy drop is more significant on our M-Type dataset, at an average of 9.2%.

• **Heterogeneity obviously slows down the FL training process in terms of both training time and training rounds.** We first analyzed the results in terms of training time. In each setting of the local training epoch, training time increases on each dataset when heterogeneity is considered. The increase ranges from 1.15× (Reddit with $E = 1$) to 2.32× (Celeba with $E = 20$) – an average of 1.74×. In addition, we find that training time increases more obviously when the number of local training epochs increases. When we set E to 20, the training time even increases by around 12 hours on Femnist and Celeba. We next analyzed the results in terms of training rounds. Similar to the training time, training rounds increase on each dataset when heterogeneity is considered. The increase ranges from 1.02× (M-Type with $E = 20$) to 2.64× (Celeba with $E = 20$) – an average of 1.42×.

Summary. *Based on the results of four datasets in various hyper-parameter settings, we observe that heterogeneity causes non-trivial performance degradation to FedAvg, including up to a* 9.2% *accuracy drop and up to* 2.32× *longer training time.*

5 RQ2: Impact on Advanced Algorithms

We now measure the impact of heterogeneity on advanced FL algorithms, including model aggregation, gradient compression, and client selection.

5.1 RQ2.1: Impact on Aggregation Algorithms

The aggregation algorithm is a key component in FL that determines how to aggregate the weights or gradients

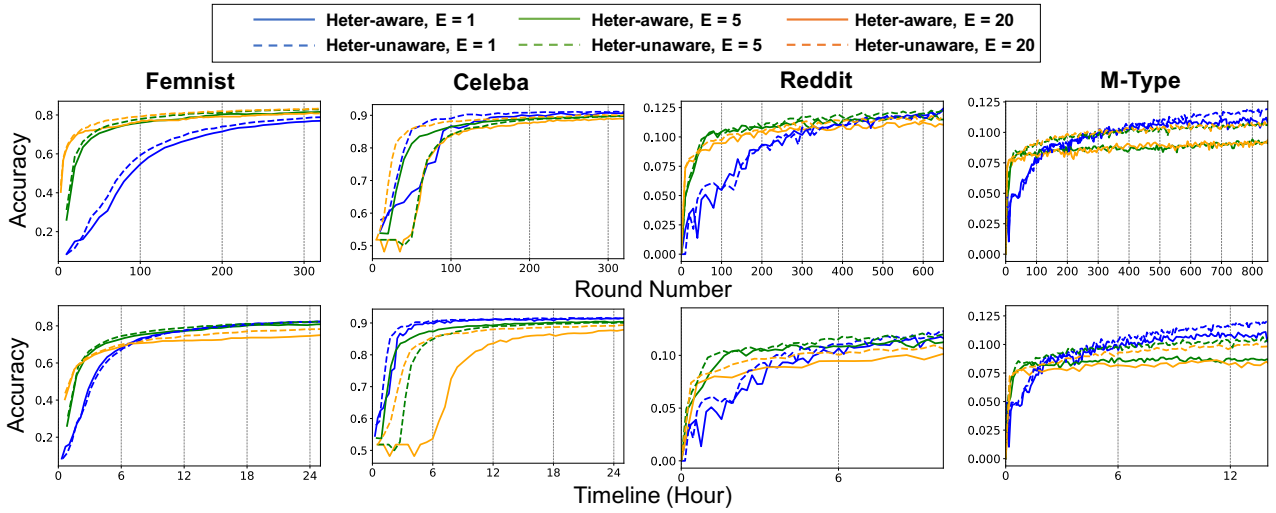


Fig. 5. The testing accuracy over time, across different numbers of local training epochs (denoted as E).

Dataset	Algo.	Acc (%) Heter-unaware	Acc (%) Heter-aware	Acc Change (ratio)	Training time Heter-unaware	Training time Heter-aware	Compression Ratio
Femnist	No Compression	84.1 (0.0%)	83.0 (0.0%)	1.2% ↓	5.56 hours (1.0×)	5.96 hours (1.0×)	100%
	Structured Updates	**84.2** (0.1% ↑)	**83.2** (0.3% ↑)	1.1% ↓	**5.23 hours (0.95×)**	**5.56 hours (0.93×)**	6.7%
	GDrop	82.2 (2.2% ↓)	81.5 (1.8% ↓)	0.8% ↓	7.17 hours (1.3×)	7.98 hours (1.3×)	21.4% ∼ 28.2%
	SignSGD	79.0 (6.1% ↓)	76.3 (8.1% ↓)	3.4% ↓	7.62 hours (1.4×)	20.5 hours (3.4×)	**3.1%**
M-Type	No Compression	9.86 (0.0%)	9.28 (0.0%)	5.9% ↓	0.54 hours (1.0×)	1.23 hours (1.0×)	100%
	Structured Updates	9.93 (0.6% ↑)	9.08 (2.2% ↓)	8.6% ↓	**0.53 hours (0.98×)**	**1.59 hours (1.3×)**	39.4%
	GDrop	8.09 (18.0% ↓)	8.27 (10.9% ↓)	2.2% ↑	5.34 hours (10.0×)	4.29 hours (3.5×)	**0.1%** ∼ **2.1%**
	SignSGD	**10.4** (6.0% ↑)	**9.55** (2.9% ↑)	8.5% ↓	1.45 hours (2.7×)	3.93 hours (3.2×)	3.1%

TABLE 4
The performance of different gradient-compression algorithms. Numbers in brackets indicate the accuracy change compared to the "No Compression" baseline. "Acc Change" refers to the accuracy change introduced by heterogeneity. The compression ratio is the fraction of the size of the compressed gradients to the original size.

uploaded from multiple devices. Besides *FedAvg*, various aggregation algorithms have been proposed to improve efficiency [17], [18], [48], ensure fairness [13], preserve privacy [36], [48], etc. To study how heterogeneity affects the performance of aggregation algorithms, we focus on two representative ones – *q-FedAvg* [13] and *FedProx* [17] – both of which are open-sourced. *q-FedAvg* is proposed to address the fairness issues in FL. It minimizes an aggregated reweighted loss so that the devices with higher loss are given higher relative weights. *FedProx* is proposed to tackle hardware heterogeneity in FL. Compared to *FedAvg*, *FedProx* allows devices to perform various amounts of training work based on their available system resources, while *FedAvg* simply drops the stragglers that fail to upload the model updates. *FedProx* also adds a proximal term to the local optimization objective (loss function) to limit the impact of variable local updates.

We use *FedAvg* as the baseline for comparison. Due to the different optimization goals of *q-FedAvg* and *FedProx*, we make the comparison separately. We show the results for *q-FedAvg* in Table 3, which illustrates the same metrics that *q-FedAvg* evaluates: variance of accuracy, worst 10% accuracy (i.e., 10% quantile of accuracy across devices), and best 10% accuracy (i.e., 90% quantile of accuracy across devices). We show the results for *FedProx* in Figure 6, which presents the accuracy changes by round. Due to space limitations, we only show the results on two datasets: one using the CNN model (Femnist) and another using the LSTM model (M-Type). Our observations are as follows.

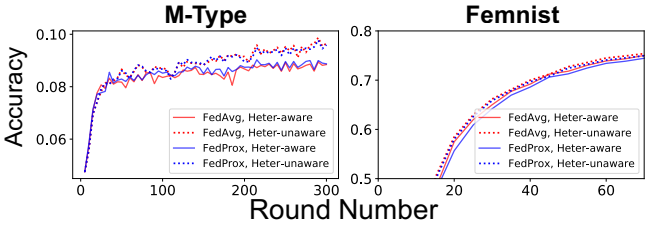


Fig. 6. The training performance of *FedProx* and *FedAvg* with and without heterogeneity.

• ***q-FedAvg*, which is supposed to address fairness issues, is less effective in ensuring fairness in heterogeneity-aware settings.** According to Table 3, the worst 10% accuracy of *q-FedAvg* under heterogeneity-unaware settings is higher than that of *FedAvg*, and *q-FedAvg* also obtains lower accuracy variance on both datasets. However, under heterogeneity-aware settings, the variance reduction decreases from 26.3% to 21.7% on Femnist and from 10.5% to 3.7% on M-Type, respectively. This is probably because *q-FedAvg* cannot tackle the bias in device selection introduced by state heterogeneity (see details in Section 7.3), which makes it less effective in ensuring fairness.

• ***FedProx* is less effective in improving the training process with heterogeneity considered.** According to Figure 6, *FedProx* only slightly outperforms *FedAvg* on M-Type,

and heterogeneity causes an accuracy drop of 7.5%. On Femnist, *FedProx* achieves the same performance as *FedAvg* in heterogeneity-unaware settings and slightly underperforms *FedAvg* in heterogeneity-aware settings. heterogeneity causes an accuracy drop of 1.2%. Note that *FedProx* incorporates hardware heterogeneity into its design while leaving state heterogeneity unsolved. We manually checked the involved devices and found that only 51.3% of devices had undergone the training when the model reached the target accuracy. As a result, the model may have been dominated by these active devices and performed badly on other devices.

Summary. *The effectiveness of two state-of-the-art FL aggregation algorithms – q-FedAvg and FedProx – is undermined by heterogeneity. Specifically, with heterogeneity considered, the accuracy variance reduction brought by q-FedAvg drops by 17.5% (from 26.3% to 21.7%); due to heterogeneity, FedProx suffers from a* $1.2\% \sim 7.5\%$ *accuracy drop and results in 48.7% of clients never contributing to the global model.*

5.2 RQ2.2: Impact on Gradient-Compression Algorithms

The cost of device-server communication is often reported as a major bottleneck in FL [2], so we investigate extensively studied gradient-compression algorithms to reduce the communication cost. Specifically, we focus on three well adopted gradient-compression algorithms: *Structured Updates* [12], *Gradient Dropping* (*GDrop*) [54], and *SignSGD* [55]. For each of them, we adjusted the hyper-parameters to achieve the highest accuracy through massive experiments. For *Structured Updates*, we set the maximum rank of the decomposed matrix to 100; for *GDrop*, we set the weight dropout threshold to 0.005; for *SignSGD*, we set the learning rate to 0.001, the momentum constant to 0, and the weight decay to 0. We used *FedAvg* with no compression as the baseline for comparison. Besides accuracy and training time/rounds, we also used the compression ratio (described in Section 3.4) as a measurement metric for these algorithms. We present the metric values of the three compression algorithms as well as the baseline in heterogeneity-unaware and heterogeneity-aware settings in Table 4. Similar to Section 5.1, we only report the results on Femnist and M-Type. We summarize our findings as follows.

• **Heterogeneity introduces a similar accuracy drop for compression algorithms as for the basic algorithm.** We measured the accuracy change introduced by heterogeneity (noted as *Acc Change* in Table 4). We observe that the introduced accuracy degradation (3.1% on average) is similar to that which we observe in Section 4 (3.2% on average). Accuracy dropped by an average of 1.7% on Femnist and 5.3% on M-Type. This is reasonable because heterogeneity will not affect the compressed gradients.

• **Gradient-compression algorithms barely speed up model convergence in heterogeneity-aware settings.** Although all these algorithms compress the gradients and reduce the communication cost significantly (the compression ratio ranges from 0.1% to 39.4%), the training time is seldom shortened (only *Structured Updates* shortens the convergence time by 0.93× at most) and is lengthened in most cases. For example, on M-Type in heterogeneity-aware environments, the training time is lengthened by 1.3× to 2.5× for all compression algorithms. The training time has not been shortened for two reasons. First, we find that communication accounts for only a small portion of the total learning time compared to on-device training. Most devices can finish downloading and uploading in less than 30 seconds for a model around 50M while spending more time (1-5 minutes with 5 epochs) on training. Second, accuracy increases slowly when the gradients are compressed and heterogeneity is introduced (see Section 4), thus taking more rounds to reach the target accuracy.

Summary. *Heterogeneity does not deteriorate the accuracy loss caused by gradient compression significantly. It results in a similar accuracy drop (*$0.8\% \sim 8.5\%$*) for compression algorithms as for the basic algorithm (*$0.5\% \sim 9.2\%$*).*

5.3 RQ2.3: Impact on Client-Selection Algorithms

The client-selection algorithm is also a key component that determines which clients to select in each round. Besides Naïve random selection, various client-selection algorithms have been proposed to accelerate training [21] and improve resource utilization [18], [57]. Intuitively, heterogeneity could affect the selection process because available clients change over time. To measure its impact, we focus on two representative algorithms: *Oort* [21] and *FedCS*. Both of them are reported to shorten training time and improve accuracy compared to random selection. *Oort* prioritizes the use of those clients who have both data that offers the greatest utility in improving model accuracy and the capability to run training quickly. *FedCS* treats client selection as a maximization problem that selects as many clients as possible under limited resources (deadline and bandwidth). *FedCS* solves the problem by proposing a heuristic algorithm based on the greedy algorithm for a maximization problem with a knapsack constraint [58].

We use random selection as the baseline and run *FedAvg* on our datasets in heterogeneity-aware and heterogeneity-unaware settings. We report training time and accuracy as the metrics given that both of them are proposed to improve FL training performance. We reused *Oort*'s open-source code and default hyper-parameters [5]. We reproduced *FedCS*'s algorithm and set the deadline to the optimal value we found in Section 6. We summarize the results in Table 5, from which we derive the following findings.

• **Heterogeneity hinders client-selection algorithms from accelerating FL training.** According to Table 5, both *Oort* and *FedCS* improve accuracy and shorten training time compared to random selection in heterogeneity-unaware settings. However, they become less effective when heterogeneity is involved. For example, in the M-Type dataset, *Oort* achieves 6.9% higher accuracy in heterogeneity-unaware settings but the accuracy increase is only 1.8% when heterogeneity is introduced. Similarly, *Oort* shortens training time by 40.5% in heterogeneity-unaware settings and the saved time reduces to 19.6% in heterogeneity-aware settings. What is worse, *FedCS* may even under-perform the baseline in heterogeneity-aware settings.

[5] *Oort* has 14 hyper-parameters that can be adjusted, which made it almost impossible to select optimal settings in our experiments. We chose to use the default ones.

Dataset	Selection Algo.	Acc (%) Heter-unaware	Acc (%) Heter-aware	Training time(H) Heter-unaware	Training time(H) Heter-aware
Femnist	Random	84.1 (0.0%)	83.0 (0.0%)	7.62 (0.0%)	9.45 (0.0%)
	Oort	85.4 (1.5% ↑)	83.2 (0.2% ↑)	4.38 (42.5% ↓)	8.83 (6.5% ↓)
	FedCS	84.8 (0.8% ↑)	82.0 (1.2% ↓)	5.25 (31.1% ↓)	9.96 (5.4% ↑)
M-Type	Random	10.51 (0.0%)	9.28 (0.0%)	3.95 (0.0%)	4.94 (0.0%)
	Oort	11.24 (6.9% ↑)	9.45 (1.8% ↑)	2.35 (40.5% ↓)	3.97 (19.6% ↓)
	FedCS	10.59 (0.8% ↑)	9.03 (2.7% ↓)	3.05 (22.8% ↓)	5.62 (13.8% ↑)

TABLE 5
The impact of heterogeneity on client-selection algorithms. Values in brackets represent the accuracy or training time change compared to the baseline.

• **Theoretical analysis.** We analyze the reasons why client-selection algorithms are hindered. The convergence speed of Stochastic Gradient Descent (SGD) can be defined as the reduction R of the divergence of model weight w from its optimal w^* in two consecutive rounds t and $t+1$ [59], [60].

$$R = \left[\|w_t - w^*\|_2^2 - \|w_{t+1} - w^*\|_2^2 \right]$$

If the learning rate of SGD is η and we use loss function L to measure the training loss between input features x and the label y, then $w_{t+1} = w_t - \eta \bigtriangledown L(w_t(x_i), y_i)$. We set the gradient $G_t = \bigtriangledown L(w_t(x_i), y_i)$ for brevity. Then R can be calculated from the former equation as:

$$\begin{aligned}
R &= -\mathbb{E}\left[(w_{t+1} - w^*)^T (w_{t+1} - w^*) - (w_t - w^*)^T (w_t - w^*)\right] \\
&= -\mathbb{E}\left[w_{t+1}^T w_{t+1} - 2w_{t+1}w^* - w_t^T w_t + 2w_t w^*\right] \\
&= -\mathbb{E}\left[(w_t - \eta G_t)^T (w_t - \eta G_t) + 2\eta G_t^T w^* - w_t^T w_t\right] \\
&= -\mathbb{E}\left[-2\eta(w_t - w^*)G_t + \eta^2 G_t^T G_t\right] \\
&= 2\eta(w_t - w^*)\mathbb{E}\left[G_t\right] - \eta^2 \mathbb{E}\left[G_t\right]^T \mathbb{E}\left[G_t\right] - \eta^2 Tr\left(\mathbb{V}\left[G_t\right]\right)
\end{aligned}$$

It has been proven that optimizing the first two terms is intractable due to their joint dependency on $\mathbb{E}[G_t]$, and client-selection algorithms, like *Oort*, are designed to gain speed over random sampling by sampling important data bins to minimize $Tr(\mathbb{V}[G_t])$ (i.e., reducing the variance of gradients while respecting the same expectation $\mathbb{E}[G_t]$) [59], [61]. However, the clients with useful data (to improve accuracy) will not always be available in heterogeneity-aware settings (we will provide more detail in Section 7.3). As a result, $Tr(\mathbb{V}[G_t])$ cannot be effectively minimized and the convergence speed slows. What is more, this selection scheme would probably make the high-end devices dominate the model, leading to a biased result. Finally, both algorithms are less effective in heterogeneity-aware settings.

Summary. *Heterogeneity hinders the state-of-the-art client-selection algorithms from improving accuracy and accelerating FL. For example, in M-Type, the accuracy increase brought by Oort dropped by 73.9% (from 6.9% to 1.8%) due to heterogeneity. Moreover, FedCS even slightly underperformed the baseline in heterogeneity-aware settings.*

6 RQ3: Impact on FL Configurations

As shown in Section 3.3, many system configurations are introduced due to heterogeneity. How to select these configurations, however, has seldom been studied in previous work due to the lack of a heterogeneity-aware FL platform along with a representative dataset (like the IMA dataset in Section 3.2). In this subsection, we show how these configurations have a non-trivial impact on the performance of the FL process and provide empirical advice on their selection.

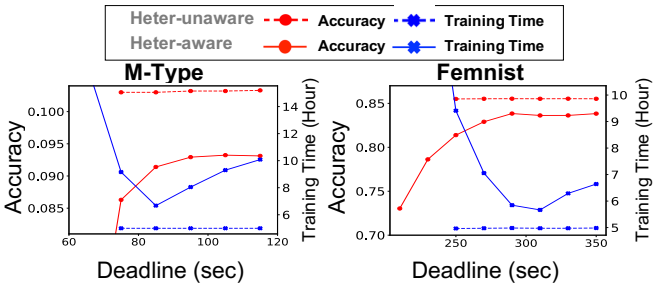


Fig. 7. The impact of reporting deadline on convergence time and accuracy.

6.1 Deadline

A deadline is set to avoid excessively long server waiting times in each round of the FL process [1]. Deadline is inherently related to heterogeneity: hardware heterogeneity will introduce varied training speeds and state heterogeneity will cause unexpected client drop-outs, so both will affect the selection of the deadline. To measure the impact of heterogeneity on deadline selection, we ran FedAvg on our datasets and set different deadlines (from too tight to too loose) in heterogeneity-aware and heterogeneity-unaware settings. We controlled other configurations and hyper-parameters to be consistent with and the same as those in previous experiments. We show our results in Figure 7 and summarize our findings below.

• **In heterogeneity-aware settings, a small change in deadline will lead to significant changes in accuracy and training time.** As illustrated in Figure 7, as the reporting deadline increases, accuracy increases at first and then remains stable. Taking Femnist as an example, accuracy rises from 0.74 to 0.84 as the deadline increases from 210 seconds to 290 seconds, and remains stable around 0.83–0.84 with an even higher reporting deadline. The reason for such a trend is that when the deadline is too tight (within a short period), a large portion of clients, especially low-end devices, can hardly (or seldom) contribute to the global training process; when the deadline is too loose (within a long period), more clients are involved in FL training but the model's capacity is reached.

• **In heterogeneity-unaware settings, one can simply choose a deadline that is loose enough and further**

lengthening it will not affect accuracy and training time. As shown in Figure 7, once the deadline is loose enough ($\geq$250s for Femnist and $\geq$80s for M-Type), accuracy and training time tend to be stable. This is because clients have a unified training speed (no hardware heterogeneity) and no clients will drop out (no state heterogeneity) [6]. Perhaps for the above reasons, the use of deadline and its influence are mostly ignored [13], [25] or considered without in-depth exploration [17], [18] in the existing FL literature. Our experimental results demonstrate that deadline is a critical configuration that should be set carefully in heterogeneity-aware settings.

• **To consider deadlines in heterogeneity-aware settings, a proper value can be set by monitoring the proportion of failed clients.** According to Figure 8, there is an optimal deadline that reaches the highest accuracy in the shortest training time (e.g., 310s for Femnist and 85s for M-Type). But despite its high impact, how to choose a proper deadline (or a proper interval) has rarely been studied in the FL literature. Thus, we present a heuristic approach to choosing the optimal reporting deadline. The proposed approach is based on an observation: the optimal deadline is correlated to the client failure rate (Section 3.4). As shown in Figure 8, for all tested datasets, the convergence time reaches the optimal (shortest) when the client failure rate is around 10%. This finding guides developers to select a proper deadline based on the monitored client failure rate with only one or several rounds. This is much more efficient compared to exploring different deadlines with end-to-end experiments (as our experiments did), each of which can take hundreds of rounds. The saved effort (e.g., experiment time in GPU-hours) can be up to two to three orders of magnitude, as estimated.

6.2 Reporting Fraction

Reporting fraction is set to control whether a round can be committed: in each round, if the proportion of successfully uploaded clients is smaller than the fraction, this round will be discarded and the global model will not be updated (fail to commit this round). It can be regarded as a trade-off between failure tolerance and fairness. A lower fraction means that more client failure is acceptable in each round (higher failure tolerance), and a higher fraction means that more clients are needed to commit this round (better fairness). Since client failure and fairness issues arise from heterogeneity, the reporting fraction is also a critical configuration. To measure the impact of heterogeneity on the selection of a reporting fraction, we ran FedAvg on our datasets and set different reporting fractions (from too small, 0.1, to too big, 1.0) in heterogeneity-aware and heterogeneity-unaware settings. We controlled for other configurations (we made sure that the deadline was long enough) and hyper-parameters to be consistent. We show our results in Table 6 and summarize our findings below.

• **Heterogeneity will exacerbate the adverse effects of an improper (lower) reporting fraction.** According to Table 6, accuracy generally decreases with lower reporting fractions in both settings because a lower fraction will lead to fewer client updates in each round. More importantly, in heterogeneity-aware settings, the accuracy drop (training time lengthened) caused by heterogeneity gets bigger when the reporting fraction decreases. For example, on M-Type, the accuracy drop increases from 11.7% to 26.4% when the fraction decreases from 0.9 to 0.1, which means that the reporting fraction becomes more influential to FL when heterogeneity is considered. The reason is that low-end devices are sensitive to the change in reporting fraction in heterogeneity-aware settings and it becomes harder to promise fairness, while devices have the same hardware capacity and get the same chance to contribute to the global model in heterogeneity-unaware settings.

• **The optimal reporting fraction is different in heterogeneity-aware and heterogeneity-unaware settings.** As shown in Table 6, the optimal reporting fraction is much easier to set in heterogeneity-unaware settings (the optimal value is 1.0), i.e., all clients upload their model updates. Remember that these clients have the same training speed and will never drop out, so setting the fraction to 1.0 will not make any rounds fail. In the existing FL literature, most work assumes that all selected clients could upload successfully [15], [16], [18], which means that the reporting fraction is set to 1.0 by default. Although our experiments verify this setting is optimal under heterogeneity-unaware settings, it could be rather different when heterogeneity is involved. By contrast, in heterogeneity-aware settings, the optimal reporting fraction is less than 1.0 (0.9 for Femnist and 0.8 for M-Type). And if we set it to 1.0, most rounds fail because client failure is almost inevitable (more detail in Section 7). Note that reporting fraction is a trade-off between failure tolerance and fairness and an improper value could lead to a severe performance decrease, so it is important to set it carefully. According to our experiments, an empirically optimal fraction is $0.8 \sim 0.9$.

Summary. *The optimal configuration is significantly different in heterogeneity-aware and heterogeneity-unaware settings. The deadline can simply be set to a value that is loose enough for all clients in heterogeneity-unaware settings (e.g., 350s in the Femnist dataset). However, in heterogeneity-aware settings, a tight (<250s) or loose (>330s) deadline is sub-optimal. A suitable deadline can be found quickly by monitoring the client failure rate (when the rate is around 10%). The reporting fraction can simply be set to 1.0 in heterogeneity-unaware settings since all clients will never fail in each round, but the optimal value is* $0.8 \sim 0.9$ *in heterogeneity-aware settings.*

7 RQ4: Influential Factors for Impact

Given the non-trivial negative impact of heterogeneity, we dive deeper to analyze the influential factors for this impact. Specifically, we first break heterogeneity down into two types to analyze their individual impact (Section 7.1). Then we report two phenomena that we have found to be particularly obvious in heterogeneity-aware settings: (1) selected devices can fail to upload their model updates, which we call *device failure* (Section 7.2); and (2) devices that succeed in uploading make a biased contribution to the global model, which we call *participant bias* (Section 7.3).

[6] Although clients could have different numbers of training samples, the FL system would usually limit the maximum sample number in each round to balance training time across clients [9].

Dataset	Fraction	Acc (%) Heter-unaware	Acc (%) Heter-aware	Training time(H) Heter-unaware	Training time(H) Heter-aware
Femnist	0.1	80.2	74.5 (7.1% ↓)	9.56	21.72 (2.27×)
	0.3	83.2	80.2 (3.6% ↓)	8.53	11.83 (1.38×)
	0.5	83.8	81.0 (3.3% ↓)	7.97	10.77 (1.35×)
	0.7	83.9	82.0 (2.3% ↓)	7.62	10.05 (1.32×)
	0.8	84.1	83.0 (1.3% ↓)	7.62	9.45 (1.24×)
	0.9	84.1	**83.1** (1.2% ↓)	7.62	**8.89** (1.17×)
	1.0	**84.3**	-	**7.62**	-
M-Type	0.1	9.53	7.01 (26.44% ↓)	6.49	12.30 (1.90×)
	0.3	9.94	8.38 (15.69% ↓)	5.75	6.92 (1.20×)
	0.5	10.23	8.64 (15.54% ↓)	5.05	6.53 (1.29×)
	0.7	10.44	8.86 (15.13% ↓)	5.33	6.32 (1.19×)
	0.8	10.44	**9.31** (10.82% ↓)	3.99	**4.78** (1.20×)
	0.9	10.51	9.28 (11.70% ↓)	3.95	4.94 (1.25×)
	1.0	**10.52**	-	**3.95**	-

TABLE 6
The performance of FedAvg with varied reporting fraction settings (0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 1.0). Values in brackets represent the accuracy drop or training time increase compared to those in heterogeneity-unaware settings (the larger the value, the greater the negative effect of heterogeneity). – means that most rounds fail. The optimal value is in **bold**.

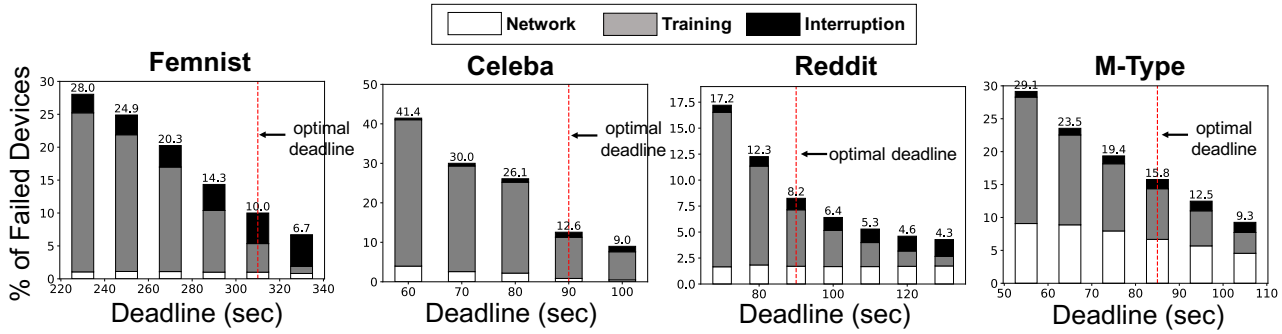


Fig. 8. The prevalence of different failure reasons. The optimal deadline (red line) refers to the one that achieves the shortest training time.

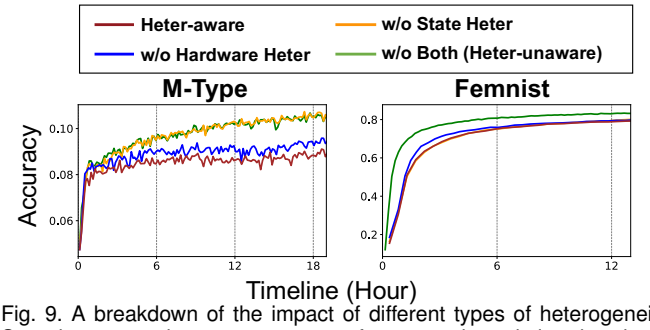


Fig. 9. A breakdown of the impact of different types of heterogeneity. State heterogeneity causes more performance degradation than hardware heterogeneity. "Heter" is short for heterogeneity.

7.1 Breakdown of Heterogeneity

The preceding results indicate the joint impact of two types of heterogeneity. To analyze their individual impact, we disabled hardware heterogeneity, i.e., all devices had the same computational and communication capacities (noted as "w/o hardware heter"). Similarly, we disabled state heterogeneity, i.e., devices were always available at any time and would not drop out (noted as "w/o state heter"). We show accuracy changes with training time in Figure 9.

• **Both state heterogeneity and hardware heterogeneity slow down model convergence.** According to Figure 9, state heterogeneity leads to a comparable increase in training time to hardware heterogeneity, i.e., 1.72× vs. 1.26× on M-Type and 2.34× vs. 2.62× on Femnist. This is reasonable because both drop-outs (introduced by state heterogeneity) and low-end devices (introduced by hardware heterogeneity) affect training time.

• **State heterogeneity has more influence on model accuracy than hardware heterogeneity.** As shown in Figure 9, state heterogeneity leads to a more significant accuracy drop than hardware heterogeneity, i.e., 9.5% vs. 0.4% on M-Type and 1.1% vs. 0.1% on Femnist. Note that existing FL-related studies usually ignore state heterogeneity (see Section 2). Our results show that state heterogeneity is more responsible for a drop in model accuracy, which explains why *FedProx* (which considers hardware heterogeneity) is less effective given both types of heterogeneity (refer to Section 5.1).

7.2 Device Failure

Device failure refers to the phenomenon that a selected device misses the deadline to upload the model updates in a round. It can slow down model convergence and waste valuable device resources (computations, energy, etc.). However, device failure has seldom been studied in prior work, probably because it is directly related to FL heterogeneity.

Heuristically, we categorize device failure with three possible causes: (1) ***Network failure*** is detected if the device takes an excessively long time (default: 3× the average) to communicate with the server due to a slow or unreliable network connection. (2) ***Interruption failure*** is detected if the device fails to upload the model updates due to user interruption, e.g., the device is taken off charge during

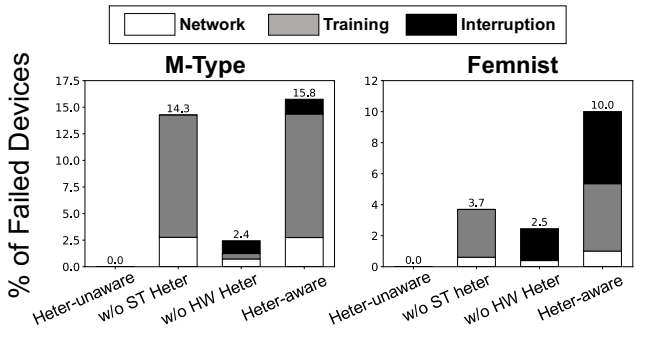


Fig. 10. The influence of different kinds of heterogeneity on device failure.

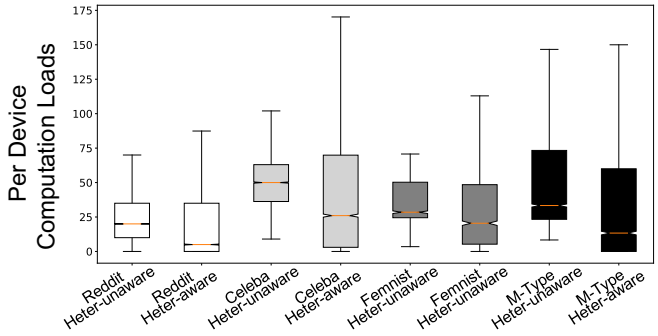


Fig. 11. The distribution of computations across devices during FL training.

training. (3) ***Training failure*** is detected when the device takes too long for the training.

To understand device failure, we zoomed into the previous experiments under varied round deadlines. We varied the deadline because we found that the proportion of failed devices was greatly affected by it. Similar to Section 7.1, we also checked hardware heterogeneity's and state heterogeneity's influence on device failure. The key questions we want to answer here are: (1) how often devices may fail and what the corresponding reasons for that failure are; and (2) which type of heterogeneity is the major factor. The results are illustrated in Figures 8 and 10, from which we make the following key observations.

• **Heterogeneity introduces non-trivial device failure even when an optimal deadline setting is given.** The overall proportion of failed devices reaches 11.6% on average, with an optimal deadline setting that achieves the shortest training time. A tight deadline increases the failure proportion because devices receive less time to finish their training tasks. We look into three types of failure and find the following. (1) Network failure accounts for a small fraction of device failure (typically less than 5%) and it is more stable than other types of failure. (2) Interruption failure is affected by the deadline but in a moderate way. We further break down interruption failure into three sub-categories corresponding to three restrictions on training [1]. Specifically, results show that the training process is interrupted by user interaction, battery charge off, and network changes, with a probability of 46.06%, 36.96%, and 17.78% respectively. (3) Training failure is heavily affected by deadline. This type of failure accounts for the majority of device failures when the deadline is too tight. Even with the optimal deadline setting, this type of failure still occurs because we observe that some low-end devices with too much local data sometimes fail to meet the deadline.

• **Hardware heterogeneity leads to more device failure than state heterogeneity.** According to Figure 10, hardware heterogeneity is more responsible for device failure than state heterogeneity. For example, on M-Type, hardware heterogeneity causes 14% of failed devices on average while state heterogeneity causes only 2.5%. This is probably because when hardware heterogeneity is considered, there are low-end devices that require a longer training time.

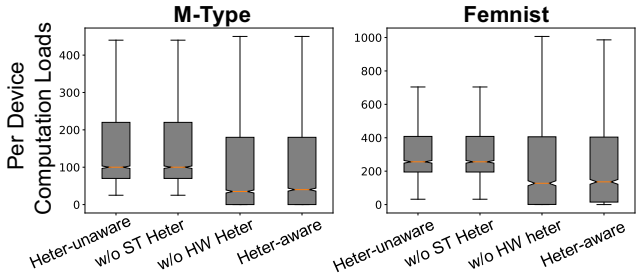


Fig. 12. A breakdown of the impact of different types of heterogeneity on participant bias.

7.3 Participant Bias

Participant bias refers to the phenomenon that devices do not participate in FL with the same probability. It can lead to different contributions to the global model, making some devices under-represented.

To measure the participant bias introduced by heterogeneity, we ran the same FL tasks as in Section 4. We took the number of computations to reflect the devices' degree of participation. Since it is difficult to compare the computation of different models directly, we divided them by the number of computations in a training epoch (noted as computation loads). Figure 11 illustrates the distribution of computation loads across devices when the global model converges. As in Section 7.1, we also break our results down to explore the impact of different types of heterogeneity. We summarize our findings as follows.

• **Computation loads get more uneven in heterogeneity-aware settings.** Variance increases by 2.4× (Reddit) to 10.7× (Femnist). Compared to heterogeneity-unaware environments, where every device participates with equal probability, computation loads tend to polarize in heterogeneity-aware environments. On Celeba, the maximum computation load increases by 1.17×.

• **The number of inactive devices increases significantly in heterogeneity-aware settings.** The median computation load dropped by 28% (Femnist) to 75% (Reddit), indicating that more inactive devices appeared. Compared to the heterogeneity-unaware environment, where the top 30% of devices contributed 54% of all computations, the top 30% of devices in heterogeneity-aware environments contributed 81% of all computations, putting the inactive devices at a

disadvantage.

• **Up to 30% of devices had not participated in the FL process when the global model reached the target accuracy in heterogeneity-aware settings.** To investigate the reasons for these inactive devices, we inspected the percentage of participating devices over time and demonstrate the result in Figure 13. We found that when the model converged (6-24 hours in our experiments), more than 30% of devices had not participated. In heterogeneity-unaware environments, the participating devices accumulated quickly and soon covered the total population in 12 hours, while in heterogeneity-aware environments, accumulation slowed a lot and it took much longer to converge (more than 48 hours).

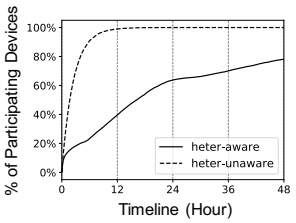


Fig. 13. Percentage of participating devices over time.

• **State heterogeneity is more responsible for participant bias.** As shown in Figure 12, state heterogeneity is the main reason for computation bias. It causes a similar computation distribution as that in heterogeneity-aware environments. This is probably because state heterogeneity introduces bias in device selection, i.e., although the server selects devices randomly, the available devices that can be selected highly depend on whether the device can meet the state criteria (see Section 3.2.1).

Summary. *State heterogeneity has more influence than hardware heterogeneity on accuracy (e.g., a drop of 9.5% vs. 0.4% in M-Type). In addition, we identify device failure and participant bias as two influential factors for the impact of heterogeneity. For device failure, 11.6% of devices would fail to upload the model updates in each round on average if no countermeasures are taken, resulting in a waste of computational resources. Excessive training time, excessive transmission time, and user interruption account for 55%, 25%, and 20% of the failures, respectively. For participant bias, the top 30% of active devices contribute 81% of all computations and up to 30% of the devices have never participated in the FL process when the global model converges, indicating that the model could be dominated by the most active devices.*

8 Implications

In this section, we discuss actionable implications for FL algorithm designers and FL system providers based on the above findings.

8.1 For FL Algorithm Designers

Taking heterogeneity into consideration. As demonstrated in our study, heterogeneity introduces a non-trivial accuracy drop and a slowdown in training in FL, and also affects the effectiveness of some proposed methods. These findings encourage FL researchers to consider heterogeneity, especially state heterogeneity. On the one hand, when designing approaches or algorithms, researchers should consider circumstances that are common in heterogeneity-aware environments but do not exist in heterogeneity-unaware environments. For example, when designing a device-selection approach, researchers should be aware that some devices can be unavailable at a given time and the server cannot select as it wants. When designing an aggregation algorithm, researchers should guarantee that the algorithm still works given inevitable device failure. On the other hand, when evaluating FL algorithms, researchers should add necessary heterogeneity settings in the experiments according to the targeted scenario. For example, if the algorithm has an additional system overhead, this may further widen the gap in training time between different devices and should be considered during the evaluation.

Reducing device failure through a "proactive alerting" technique. In Section 7.2, we find that around 10% of devices fail to upload their model updates in typical settings. The reasons include excessive training time, an unstable network connection, and device drop-out caused by state changes. Existing efforts have explored dynamic deadlines [43] and tolerating partial work [17] to handle device failure. However, these algorithms are inadequate to handle failure caused by unstable network connections and drop-outs because they are highly dependent on the devices' states. A "proactive alerting" technique can be explored by predicting devices' future states and network conditions based on historical data. The server should assign low priority to the devices that are likely to drop out. In this way, overall device failure can be reduced and more updates can be aggregated, thus saving the hardware resources and accelerating the learning process.

Resolving bias in device selection. In Section 7.3, we find that the global model is dominated by some active devices (the top 30% of devices may contribute 81% of the total computation). The reason is that, due to state heterogeneity, devices do not participate in the learning process with the same probability even when they are randomly selected, and some (more than 30% in our experiments) have never participated when the model reaches a local optimum. To alleviate bias in device selection, a naive approach is to set a participation time window (e.g., one day) and omit the devices that have participated in this window. "Fairness" is guaranteed, but this may remarkably increase the training time of an FL task, and the length of the time window should be carefully adjusted. Adjusting the local objective (loss function) or re-weighting updates may serve as alternatives.

8.2 For FL System Providers

Building heterogeneity-aware platforms. Our results show that a heterogeneity-aware platform is necessary for developers to precisely understand how their model will perform in real-world settings. However, existing platforms [25], [62], [63], [64], [65] fail to incorporate heterogeneity into their design. Our work provides a reference implementation and can be integrated easily into these FL platforms. We also encourage system providers to collect data that fit different scenarios to further help the FL community.

Optimizing on-device training time instead of compression in unmetered (e.g., WiFi) networks. In Section 5.2, we find that gradient-compression algorithms barely speed up model convergence. The time spent on communication is relatively small in WiFi environments compared to the

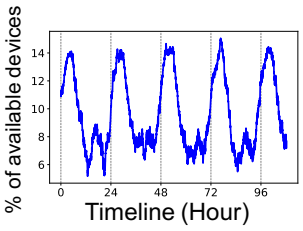


Fig. 14. Percentage of available devices over time.

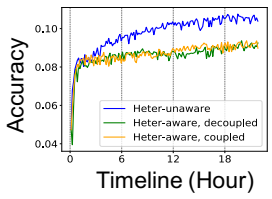


Fig. 15. Decoupling is verified on M-Type.

time spent on training. As a result, an orthogonal way to accelerate FL is to optimize the on-device training time. Possible solutions include neural architecture search (NAS) [66], [67] and using hardware AI accelerators like mobile GPU and digital signal processors (DSPs).

9 Discussion

We next discuss open problems along with the generalizability of our study.

Bias of our IMA dataset. The device state traces (Section 3.2.1) are collected from our IMA dataset (app-specific), whose users mainly reside in Southeast Asia and Latin America (geo-specific). The traces may not be fully representative of other FL scenarios. However, we believe that our findings are still faithful because (1) FL tasks are always app-specific, and improving IMA experience is a key scenario of FL [1], [9], [10]; and (2) our traces are large enough to cover the general state change patterns of smartphones. What is more, the patterns are consistent with prior work [9], as previously mentioned. Furthermore, new user traces can be seamlessly plugged into our platform, where researchers can reproduce all experiments mentioned in this paper.

Consistency with results reported by real-world FL systems. As with all existing FL platforms [25], [62], [63], [64], [65], our platform (Section 3.3) performs FL tasks in a simulation way. We carefully designed the platform to simulate real-world FL systems by considering heterogeneity. However, we acknowledge that a gap may still exist for unexpected FL glitches, e.g., software failure. We plan to further validate our platform with real-world deployment. Nevertheless, the observed patterns from our platform, e.g., device availability (Figure 14) and failure proportion (Figure 8), are consistent with the results reported from a large-scale FL deployment by Google [1]. Therefore, we believe that our findings are still valid.

Validity of randomly assigning state traces and training data to devices. In practice, heterogeneity is inherently coupled with non-IID data distribution [2]. In this study, we decouple heterogeneity from data distribution (i.e., randomly assigning a state trace to each device) to generalize our traces to other benchmark datasets. We use M-Type to verify this design because it shares the same user population with our traces. According to Figure 15, the gap between the coupled case and the decoupled case is trivial compared to the gap between the heterogeneity-unaware and heterogeneity-aware settings. This justifies our design to decouple heterogeneity from any third-party datasets.

More delicate characterization of heterogeneity. In the real world, the severity of heterogeneity varies across populations/platforms/applications. In our settings, we choose "Heter-aware" and "Heter-unaware", which represent the two ends of the heterogeneity spectrum, to demonstrate its impact. Meanwhile, our dataset also makes it possible to support a more delicate characterization. With our dataset, FL developers can easily remove/duplicate high-end/low-end or active/inactive devices to better reflect the target population. As a result, by controlling the proportion of different types of devices, it is possible to characterize heterogeneity more delicately.

Other types of heterogeneity. In this paper, we focus on the impact of hardware and state heterogeneity, but there are also other types of heterogeneity in FL. One is data heterogeneity [2], [8], which resides in the skewed and unbalanced local data distribution (non-IID data distribution) across devices. Data heterogeneity is one of the basic assumptions in FL and existing work [7], [20], [68] has conducted in-depth research on it. Since the benchmark datasets used in our experiments are all non-IID datasets, data heterogeneity is inherently considered in our study. Other types of heterogeneity [2], like software or platform, are highly relevant to the implementation of an FL system and hard to generalize. We leave those for future work to explore.

10 Conclusion and Future Work

We have collected large-scale real-world data and conducted extensive experiments to anatomize the impact of heterogeneity. Results show that (1) heterogeneity causes non-trivial performance degradation in FL tasks – up to a 9.2% accuracy drop and 2.32× slowdown in convergence; (2) recent advanced FL algorithms can be compromised and rethought with heterogeneity considered; (3) the setting of key FL configurations (deadline and reporting fraction) can be significantly different in heterogeneity-aware and heterogeneity-unaware settings; (4) state heterogeneity, which is usually ignored in existing studies, is more responsible for the aforementioned performance degradation than hardware heterogeneity; and (5) device failure and participant bias are two potential impact factors for performance degradation. These results suggest that heterogeneity should be taken into consideration in further research and that optimizations to mitigate the negative impact of heterogeneity are promising.

In the future, we plan to enhance FLASH in the following aspects: (1) *FL attacks and countermeasures*. There have been plenty of works [14], [38] that successfully attack FL through data poisoning or Byzantine fault. Correspondingly, a few countermeasures (e.g., secure aggregation [36]) are proposed to further enhance the data privacy in FL. The impact of heterogeneity on those approaches could be studied in the future. (2) *Training efficiency*. FLASH currently carries out the per-client model training in a sequential manner. To speed up the simulation, we plan to add in-parallel training support on multi-GPU or multi-server settings.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under the grant number 2020YFB2104100, the NSFC under grant numbers 62172008 and 62102045, and National Natural Science Fund for the Excellent Young Scientists Fund Program (Overseas). Zhenpeng Chen's work was supported by the ERC Advanced Fellowship under grant number 741278 (EPIC: Evolutionary Program Improvement Collaborators). Chengxu Yang and Mengwei Xu contributed equally to the work. Contact the corresponding author Xuanzhe Liu via xzl@pku.edu.cn.

REFERENCES

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

[2] 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," *arXiv preprint arXiv:1912.04977*, 2019.

[3] Z. Chen, Y. Cao, Y. Liu, H. Wang, T. Xie, and X. Liu, "A comprehensive study on challenges in deploying deep learning based software," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*, 2020, pp. 750–762.

[4] Z. Chen, H. Yao, Y. Lou, Y. Cao, Y. Liu, H. Wang, and X. Liu, "An empirical study on deployment faults of deep learning based mobile applications," in *Proceedings of the 43rd International Conference on Software Engineering, ICSE 2021*, 2021, pp. 674–685.

[5] Wikipedia, "General data protection regulation," https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, 2020, accessed Feb 5, 2020.

[6] Wiki, "California consumer privacy act," https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act, 2020, accessed Feb 5, 2020.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017, pp. 1273–1282.

[8] 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.

[9] 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," *arXiv preprint arXiv:1812.02903*, 2018.

[10] 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," *arXiv preprint arXiv:1811.03604*, 2018.

[11] 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 *International workshop on machine learning in medical imaging*. Springer, 2019, pp. 133–141.

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

[13] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proceedings of 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[15] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.

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

[17] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems 2020, MLSys 2020*, 2020.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proceedings of ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[19] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui, "Device heterogeneity in federated learning: a superquantile approach," *arXiv preprint arXiv:2002.11223*, 2020.

[20] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning," in *Proceedings of 2019 USENIX Conference on Operational Machine Learning (OpML 19)*, 2019, pp. 19–21.

[21] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021, pp. 19–35.

[22] PushShift.io, "Reddit dataset," https://files.pushshift.io/reddit/, 2020, accessed May 22, 2020.

[23] T. C. U. o. H. K. Multimedia Laboratory, "Large-scale celebfaces attributes (celeba) dataset," http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html, 2020, accessed May 22, 2020.

[24] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: extending mnist to handwritten letters," in *Proceedings of 2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2921–2926.

[25] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: a benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[26] 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 *Proceedings of the Web Conference 2021, WWW 2021*, 2021, pp. 935–946.

[27] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2020.

[28] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[29] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Billion-scale federated learning on mobile clients: a submodel design with tunable privacy," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[30] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

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

[32] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: a communication-efficient federated learning method with periodic averaging and quantization," *arXiv preprint arXiv:1909.13014*, 2019.

[33] K. Bonawitz, F. Salehi, J. Konečnỳ, B. McMahan, and M. Gruteser, "Federated learning with autotuned communication-efficient secure aggregation," *arXiv preprint arXiv:1912.00131*, 2019.

[34] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang, "Hierarchical federated learning through lan-wan orchestration," *arXiv preprint arXiv:2010.11612*, 2020.

[35] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1234–1242.

[36] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggre-

gation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[37] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv preprint arXiv:1710.06963*, 2017.

[38] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proceedings of 2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[39] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[40] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[41] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[42] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 493–506.

[43] L. Li, H. Xiong, Z. Guo, J. Wang, and C.-Z. Xu, "Smartpc: hierarchical pace control in real-time federated learning system," in *Proceedings of 2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019, pp. 406–418.

[44] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2020.

[45] H. Jiang, Z. Xiao, Z. Li, J. Xu, F. Zeng, and D. Wang, "An energy-efficient framework for internet of things underlaying heterogeneous small cell networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 31–43, 2020.

[46] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Transactions on Mobile Computing*, 2020.

[47] D. R. Brendan McMahan, "Federated learning: collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017, published April 6, 2017.

[48] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Secure federated submodel learning," *arXiv preprint arXiv:1911.02254*, 2019.

[49] J. Kang, S. Seo, and J. W. Hong, "Usage pattern analysis of smartphones," in *Proceedings of 2011 13th Asia-Pacific Network Operations and Management Symposium*, 2011, pp. 1–8.

[50] S. Yogesh, S. Abha, and S. Priyanka, "Short communication mobile usage and sleep patterns among medical students," *Indian J Physiol Pharmacol*, vol. 58, no. 1, pp. 100–103, 2014.

[51] L. L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, and Y. Liu, "nn-meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 81–93.

[52] MNN, "Training framework usage," https://www.yuque.com/mnn/cn/kgd9hd?translate=en, 2020, accessed Apr 22, 2020.

[53] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "Ai benchmark: running deep neural networks on android smartphones," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 288–314.

[54] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa, and S. Riedel, Eds., 2017, pp. 440–445.

[55] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signsgd with majority vote is communication efficient and fault tolerant," in *Proceedings of 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[56] Wikipedia, "Compression ratio," https://en.wikipedia.org/wiki/Compression_ratio, 2020, accessed Oct 18, 2020.

[57] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, and X. Wang, "Resource-efficient and convergence-preserving online participant selection in federated learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 606–616.

[58] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[59] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *International conference on machine learning*. PMLR, 2018, pp. 2525–2534.

[60] P. Zhao and T. Zhang, "Stochastic optimization with importance sampling for regularized loss minimization," in *international conference on machine learning*. PMLR, 2015, pp. 1–9.

[61] T. B. Johnson and C. Guestrin, "Training deep models faster with robust, approximate importance sampling," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[62] Tensorflow, "Tensorflow federated: Machine learning on decentralized data," https://www.tensorflow.org/federated, 2020, accessed Jan 28, 2020.

[63] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

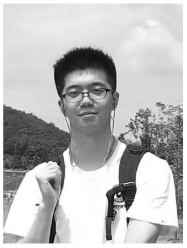
[64] PaddlePaddle, "Federated deep learning in paddlepaddle," https://github.com/PaddlePaddle/PaddleFL, 2020, accessed Jan 28, 2020.

[65] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen *et al.*, "Fedml: a research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.

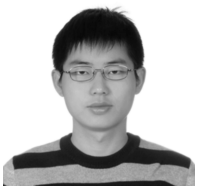
[66] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: a survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[67] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

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



Chengxu Yang is a Ph.D. candidate in the Department of Computer Science at Peking University. He received his B.S. in 2019 from the Department of Computer Science and Technology at Peking University. His main research interests lie in data privacy, including privacy protection in ML (e.g., federated learning) and compliance checking on big data platforms. Web page: https://lh-ycx.github.io.



Mengwei Xu is an Assistant Professor in the Computer Science Department at Beijing University of Posts and Telecommunications. He received his B.S. in 2015 and Ph.D. in 2020 from the Department of Computer Science and Technology at Peking University. His research interests cover the broad areas of mobile computing, edge computing, artificial intelligence, and system software. Web page: https://xumengwei.github.io.

Qipeng Wang is a Ph.D. candidate in the Department of Computer Science at Peking University. He received his B.S. in 2020 from the School of Software at Beihang University. His main research interests lie in edge intelligence and on-device learning. Web page: https://qipengwang.github.io.

Zhenpeng Chen is a Research Fellow in the Department of Computer Science at University College London. He received his B.S. in 2016 and Ph.D. in 2021 from the Department of Computer Science and Technology at Peking University. His main research interests lie in Software Engineering and Artificial Intelligence. He received the Best Full Paper Award at The Web Conference (WWW) in 2019. Web page: https://chenzhenpeng18.github.io.

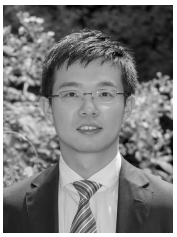
Kang Huang received his Ph.D. from Beihang University. He is the co-founder and CEO of Linggui Tech. His main research fields are artificial intelligence and machine learning.

Yun Ma is an Assistant Professor at the Institute for Artificial Intelligence, Peking University, Beijing, China. He received his B.S. in 2011 and Ph.D. in 2017 from the Department of Computer Science and Technology at Peking University. His research interests include web systems and mobile computing. He is a member of the IEEE. Web page: http://sei.pku.edu.cn/~mayun11/index.htm.

Kaigui Bian is an Associate Professor in the School of Computer Science at Peking University. He received his B.S. in Computer Science from Peking University in 2005, and his Ph.D. in Computer Engineering from Virginia Tech in 2011. He was a Visiting Young Faculty with Microsoft Research Asia in 2013. His research interests include wireless networking and mobile computing. He has received the best paper awards at five international conferences and received the IEEE Communication Society Asia–Pacific Board (APB) Outstanding Young Researcher Award in 2018. He currently serves as an Editor for IEEE Transactions on Vehicular Technology. Web page: https://kevinbkg.github.io/kg.github.io/.

Gang Huang is a full professor in the School of Computer Science, Peking University. His research interests include operating systems and middleware for Internet computing including cloud computing, mobile computing, big data, and blockchain. He is a member of the IEEE. Web page: http://sei.pku.edu.cn/~huanggang/index_en.htm.

Yunxin Liu is a Guoqiang Professor at the Institute for AI Industry Research (AIR) at Tsinghua University. He was a Principal Research Manager at Microsoft Research Asia (MSRA). He received his Ph.D., M.S., and B.S. degrees from Shanghai Jiao Tong University (SJTU), Tsinghua University, and the University of Science and Technology of China (USTC), respectively. His research interests are mobile computing and edge computing. He received the MobiSys 2021 Best Paper Award, SenSys 2018 Best Paper Runner-up Award, MobiCom 2015 Best Demo Award, and PhoneSense 2011 Best Paper Award. Web page: https://yunxinliu.github.io/.



Xin Jin is an Associate Professor in the School of Computer Science at Peking University. He received his B.S. in Computer Science from Peking University in 2011, and his Ph.D. in Computer Science from Princeton University in 2016. His research is on computer networks and computer systems, with a focus on software-defined data centers, programmable networks, and cloud computing. Web page: https://xinjin.github.io/.

Xuanzhe Liu is an Associate Professor (with tenure) in the School of Computer Science at Peking University. His research interests mainly fall in service-based software engineering and systems. Most of his recent efforts have been published at prestigious conferences including WWW, ICSE, FSE, SIGCOMM, NSDI, MobiCom, MobiSys, SIGMETRICS, and IMC, and in journals including ACM TOSEM/TOIS/-TOIT/TWEB and IEEE TSE/TMC/TSC. He is a senior member of the IEEE and the ACM, and a distinguished member of the CCF. He serves as the corresponding author of this paper. Web page: http://www.liuxuanzhe.com/.