Characterizing Impacts of Heterogeneity in Federated Learning upon Large-Scale Smartphone Data

Chengxu Yang, Qipeng Wang
Key Lab of High Confidence Software Technologies (Peking University), MoE
Beijing, China
yangchengxu@pku.edu.cn
wangqipeng@stu.pku.edu.cn

Mengwei Xu
State Key Laboratory of Networking and Switching Technology (BUPT)
Key Lab of High Confidence Software Technologies (Peking University), MoE
Beijing, China
mwx@bupt.edu.cn

Zhenpeng Chen
Key Lab of High Confidence Software Technologies (Peking University), MoE
Beijing, China
czp@pku.edu.cn

Kaigui Bian
Department of Computer Science and Technology, Peking University
Beijing, China
bkg@pku.edu.cn

Yunxin Liu
Microsoft Research
Beijing, China
yunxin.liu@microsoft.com

Xuanzhe Liu*
Key Lab of High Confidence Software Technologies (Peking University), MoE
Beijing, China
xzl@pku.edu.cn

ABSTRACT

Federated learning (FL) is an emerging, privacy-preserving machine learning paradigm, drawing tremendous attention in both academia and industry. A unique characteristic of FL is *heterogeneity*, which resides in the various hardware specifications and dynamic states across the participating devices. Theoretically, heterogeneity can exert a huge influence on the FL training process, e.g., causing a device unavailable for training or unable to upload its model updates. Unfortunately, these impacts have never been systematically studied and quantified in existing FL literature.

In this paper, we carry out the first empirical study to characterize the impacts of heterogeneity in FL. We collect large-scale data from 136k smartphones that can faithfully reflect heterogeneity in real-world settings. We also build a *heterogeneity-aware* FL platform that complies with the standard FL protocol but with heterogeneity in consideration. Based on the data and the platform, we conduct extensive experiments to compare the performance of state-of-the-art FL algorithms under *heterogeneity-aware* and *heterogeneity-unaware* settings. Results show that heterogeneity causes non-trivial performance degradation in FL, including up to 9.2% accuracy drop, 2.32× lengthened training time, and undermined fairness. Furthermore, we analyze potential impact factors and find that *device failure* and *participant bias* are two potential factors for performance degradation. Our study provides insightful implications for FL practitioners. On the one hand, our findings suggest that FL algorithm designers consider necessary heterogeneity during the evaluation. On the other hand, our findings urge system providers to design specific mechanisms to mitigate the impacts of heterogeneity.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**; • **Computing methodologies → Artificial intelligence**; • **Security and privacy → Privacy protections**.

KEYWORDS

Federated learning; heterogeneity; measurement study

ACM Reference Format:
Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. 2021. Characterizing Impacts of Heterogeneity in Federated Learning upon Large-Scale Smartphone Data. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449851

1 INTRODUCTION

In the past few years, we have witnessed the increase of machine learning (ML) applications deployed on mobile devices [11, 13, 49, 50, 52]. These applications usually need to collect personal user data to train ML models. However, due to the increasing concerns of user privacy, e.g., the recent released GDPR [47] and CCPA [45], personal data cannot be arbitrarily collected and used without permission granted [33]. Therefore, various privacy-preserving ML techniques have been proposed [2, 12, 30], where the emerging federated learning (FL) has drawn tremendous attentions [27, 28]. The key idea of FL is to employ a set of personal mobile devices, e.g., smartphones, to train an ML model collaboratively under the orchestration of a central parameter server. Since the training process of FL takes place on mobile devices (i.e., on-device training) without uploading user personal data outside devices, it is considered to be quite promising for preserving user privacy in ML applications.

Recently, various emerging FL algorithms have been proposed, e.g., *FedAvg* [30], *Structured Updates* [24], and *q-FedAvg* [29]. To

[*] Corresponding author: Xuanzhe Liu (xzl@pku.edu.cn).

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449851

evaluate these algorithms, existing FL studies typically take a simulation approach [3, 10, 29, 30] given the high cost of real deployment. However, they have no data to describe how devices participate in FL[1]. As a result, they usually have an overly ideal assumption, i.e., all the devices are always available for training and equipped with *homogeneous* hardware specifications (e.g., the same CPU and RAM capacity) [3, 20, 24, 29, 30, 34].

However, these assumptions could be too ideal for FL deployment in practice. More specifically, FL usually requires a substantial number of devices to collaboratively accomplish a learning task, which poses a unique challenge, namely **heterogeneity** [27]. In practice, the heterogeneity can be attributed to two major aspects: (1) One is from hardware specifications of devices (called *hardware heterogeneity*), e.g., different CPU, RAM, and battery life. (2) Additionally, the state and running environment of participating devices can 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 (called *device failure*), which can definitely affect the training time to obtain a converged global model. Furthermore, devices that seldom participate in an FL task due to abnormal states, e.g., CPU busy, can be underrepresented by the global model.

Although some recent studies [9, 25, 28, 37] have realized the heterogeneity in FL, its impacts have never been comprehensively quantified over large-scale real-world data. In this paper, we present the first empirical study to demystify the impacts of heterogeneity in FL tasks. To this end, we develop a holistic platform that complies with the standard and widely-adopted FL protocol [5, 17, 53], but for the first time, facilitates reproducing existing FL algorithms under *heterogeneity-aware* settings, i.e., devices have dynamic states and various hardware capacities. Undoubtedly, conducting such a study requires the data that can faithfully reflect the heterogeneity in real-world settings. Therefore, we collect the device hardware specifications and regular state changes (including the states related to device check-in and drop-out) of 136k smartphones in one week through a commodity input method app (IMA). We then plug the data into our *heterogeneity-aware* platform to simulate the device state dynamics and hardware capacity.

Based on the data and platform, we conduct extensive measurement experiments to compare the state-of-the-art FL algorithms' performance, including model accuracy and training time under heterogeneity-aware and heterogeneity-unaware settings. We select four typical FL tasks: two image classification tasks and two natural language processing tasks. For every single task, we employ a benchmark dataset for model training. Three of the benchmark datasets [14, 35, 40] have been widely used in existing FL-related studies [8, 24, 28–30], and the last one is a real-world text input dataset collected from the aforementioned IMA.

• **Findings.** Heterogeneity leads to non-trivial impacts on the performance of FL algorithms, including accuracy drop, increased training time, and undermined fairness. For the basic algorithm (§4.1), i.e., *FedAvg* [30], when heterogeneity is considered, its performance is compromised in terms of 3.1% accuracy drop (up to 9.2%) and 1.74× training time (up to 2.32×) on average. For other advanced algorithms (§4.2), i.e., gradient compression algorithms, including *Structured Updates* [24], *Gradient Dropping* [1], and *SignSGD* [4], and advanced aggregation algorithms, i.e., *q-FedAvg* [29] and *FedProx* [28], optimizations are not always effective as reported. For example, heterogeneity hinders *q-FedAvg* from addressing the fairness issues in FL. We also find that current gradient compression algorithms can hardly speed up FL convergence under heterogeneity-aware settings. In the worst case, the training time is lengthened by 3.5×. These findings indicate that heterogeneity cannot be simply ignored when designing FL algorithms.

• **Analysis of Potential Impact Factors**. We first break down the heterogeneity to analyze the individual impacts of state heterogeneity and hardware heterogeneity, respectively (§5.1). We find that both types of heterogeneity can slow down the learning process, while state heterogeneity is often more responsible for the accuracy degradation. Then we zoom into our experiments and find out two major factors that are particularly obvious under heterogeneity-aware settings. (1) *Device failure* (§5.2): On average, 11.6% of selected devices fail to upload their model updates per round due to unreliable network, excessive training time, and drop-out caused by user interruption. This failure slows down the model convergence and wastes valuable hardware resources. (2) *Participant bias* (§5.3): Devices attend 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 (top 30% devices contribute to 81% total computation). State heterogeneity is the major cause for the participant bias.

Our extensive experiments provide several insightful implications as summarized in §6. For instance, FL algorithm designers should consider necessary heterogeneity in the evaluation environment of FL, while *FL system providers* should design specific mechanisms to mitigate the impacts of heterogeneity. In summary, the major contributions of this paper are as follows:

- We build a heterogeneity-aware FL platform with a large-scale dataset collected from 136k smartphones, which can help simulate the state and hardware heterogeneity for exploring FL in real-world practice[2].
- We conduct extensive measurement experiments to demystify the non-trivial impacts of heterogeneity in existing FL algorithms.
- We make an in-depth analysis of possible factors for impacts introduced by heterogeneity. Our results can provide insightful and actionable implications for the research community.

2 BACKGROUND AND RELATED WORK

FL is an emerging privacy-preserving learning paradigm. Among different FL scenarios [21], we focus on a widely studied one, i.e., *cross-device* FL, which utilizes a federation of *client devices*[3], coordinated by a *central server*, to train a global ML model. A typical FL workflow [5] consists of many 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

[1] Although Google has built a practical FL system [5], its detailed data are not disclosed.

[2] We have released our dataset and source code at https://github.com/PKU-Chengxu/FLASH to facilitate future FL research.

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

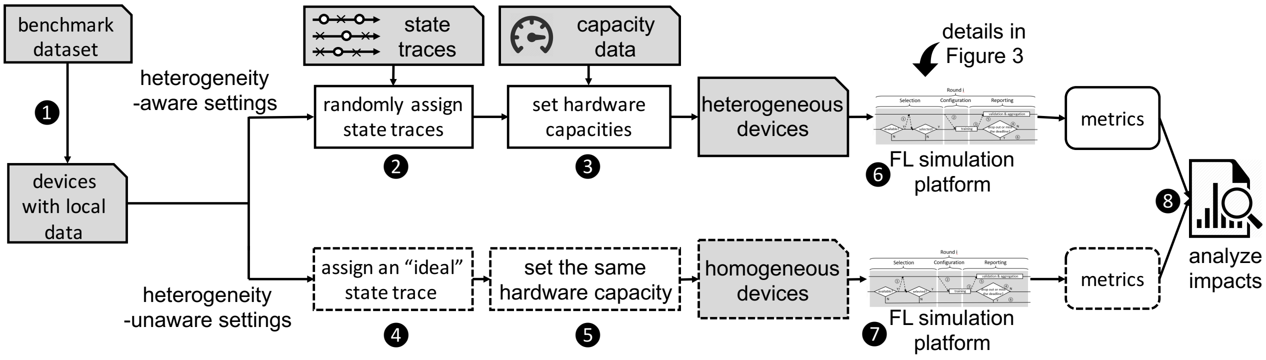

Figure 1: Overview of our methodology.

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* [30]. In *FedAvg*, devices perform multiple local training epochs, where per round a device updates the weights of its local model using its local data. Then the central server averages the updated weights of local models as the new weights of the global model. *FedAvg* is a representative FL algorithm that has been widely used in existing FL-related studies [20, 24, 29, 34] and also deployed in the industry, e.g., in Google's production FL system [9]. Therefore, we use *FedAvg* as the basic FL algorithm to study heterogeneity's impacts (§4.1).

In addition, many advanced algorithms have been proposed to optimize FL, including reducing the communication cost between the central server and devices [7, 10, 24, 41, 43, 55], enhancing privacy guarantee [3, 6, 31, 32, 36], ensuring fairness across devices [20, 29, 34], minimizing on-device energy cost [26], etc. However, most of them have not been well evaluated in a heterogeneity-aware environment, making their benefits unclear in real-world deployment. Therefore, we make the first attempt to study how heterogeneity impacts the effectiveness of these advanced FL algorithms (§4.2).

Heterogeneity is considered as one of the core challenges in FL [27]. Some existing work [9, 25, 28, 37] has studied the heterogeneity in FL but not in a comprehensive way. In particular, they ignore state heterogeneity and randomly set training time to simulate hardware heterogeneity, leaving communication capacities unconsidered. For example, *FedProx* [28] handles the 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 state changes of devices remain unconsidered. *FedCS* [37] accelerates FL by managing devices based on their resource conditions and allowing the server to aggregate as many device updates as possible, but it assumes that the network is stable and not congested, and randomly sets the training time from 5 to 500 seconds. Chai et al. [9] have studied the impacts of hardware heterogeneity by allocating varied CPU resources when simulating FL, but they leave state heterogeneity unconsidered. Our study differs from existing work in two aspects: (1) we comprehensively consider hardware heterogeneity and state heterogeneity in the experimental environment powered by real-world data, and (2) we build an FL scenario with a much larger device population (up to 136k).

3 THE MEASUREMENT APPROACH

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 their own local data for training (❶). For a fair comparison, we always use the same partition strategy in the heterogeneity-aware settings and heterogeneity-unaware settings, i.e., the local training data on a given device are the same.

For heterogeneity-aware settings, we randomly assign 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 are collected from large-scale real-world mobile devices through an IMA app (details in §3.2). As a result, we get a heterogeneous device set with different local training data, hardware capacities, and state change dynamics.

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

We next deploy the two device sets to our FL simulation platform and execute the FL task (e.g., image classification) under the same configurations (❻ and ❼). The simulation platform extends the standard FL protocol with heterogeneity consideration, e.g., a device can quit training due to a state change (details in §3.3). We finally analyze heterogeneity's impacts by comparing the metric values achieved by heterogeneous devices and homogeneous devices (❽).

3.2 The Datasets

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

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 at 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.

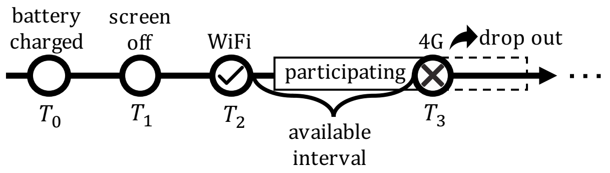

Figure 2: A trace is a series of state changes over time.

3.2.1 ***IMA dataset.*** To power the heterogeneity-aware settings, we collect 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 within one week starting from Jan. 31 in 2020. More specifically, every time the aforementioned state changes, the IMA records it with the timestamp and saves it as a state entry (refer to Table 1).

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

The state traces are to determine the time intervals when a device is available for local training, which are critical in understanding the FL performance under 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 [5], i.e., when a device is idle, charged, and connected to WiFi. 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.

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 the datasets covering a small group of devices [22, 54].

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

For the computational capacity, we seek to obtain the training speed for each given device. However, our collected IMA dataset contains more than one thousand types of devices, making it rather difficult to profile. Thus, we employ a "clustering" approach by mapping all device models to a small number of representative device models that we afford to offline profile. The mapping consists of two steps: (1) The total device models are first mapped to the device models profiled by AI-Benchmark [19], a comprehensive AI performance benchmark. For a few device models that AI-Benchmark does not cover, we make a random mapping. It reduces the number of device models to 296. (2) The remaining device models are then mapped to what we afford to profile. So far, we have profiled three representative and widely-used device models (Samsung Note 10, Redmi Note 8, and Nexus 6), and we plan to include more device models in the future. To profile these devices, we run on-device training using the open-source ML library DL4J [15] and record their training time for each ML model used in our experiments. We are aware of learning-based approaches [51] to obtain the on-device ML performance, but our empirical efforts show that these approaches are not precise enough for on-device training tasks.

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

3.2.2 ***Benchmark datasets.*** We use four benchmark datasets to quantitatively study the impacts of heterogeneity on FL performance. Three of them (i.e., Reddit [40], Femnist [14], and Celeba [35]) are synthetic datasets widely adopted in FL literature [3, 27, 29, 37], while the remaining one is a real-world input corpus collected from our IMA, named as M-Type. M-Type contains texts input from the devices covered in the state traces in §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 use CNN models, and for Reddit and M-Type, we use LSTM models. The four models are implemented by *Leaf* [8], a popular FL benchmark. All the datasets are non-IID datasets, i.e., the data distribution is skewed and unbalanced across devices, which is a common data distribution in FL scenarios [21]. We randomly split the data in each device into a training/testing set (80%/20%).

3.2.3 ***Ethic considerations.*** All the data are collected with explicit agreements with users on user-term statements and a strict policy in data collection, transmission, and storage. The IMA users are given an explicit option to opt-out of having their data collected. In addition, we take very careful steps to protect user privacy and preserve the ethics of research. First, our work is approved by the Research Ethical Committee of the institutes that the authors are currently affiliated with. Second, the users' identifies are all completely anonymized during the study. Third, the data are stored and processed on a private, HIPPA-compliant cloud server, with strict access authorized by the company that develops the IMA. The whole process is compliant with the privacy policy of the company.

3.3 The Simulation Platform

Our platform follows the standard FL protocol [5] and divides the simulation into three main following phases as shown in Figure 3. We also follow the Google's report [53] to configure the FL systems , e.g., the time that the server waits for devices to check-in. Given

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

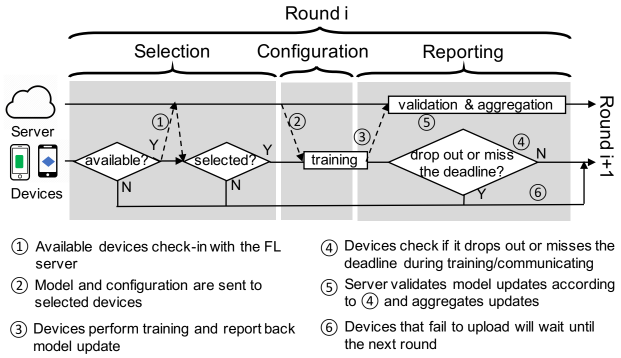
Round i
Selection
Configuration
Reporting
Server
Devices
validation & aggregation
Round i+1
①
②
③
④
⑤
⑥
available?
selected?
training
drop out or miss the deadline?
Y
N
① Available devices check-in with the FL server
② Model and configuration are sent to selected devices
③ Devices perform training and report back model update
④ Devices check if it drops out or misses the deadline during training/communicating
⑤ Server validates model updates according to ④ and aggregates updates
⑥ Devices that fail to upload will wait until the next round

Figure 3: We build our FL simulation platform atop the standard FL protocol [5].

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). The configuration is used to instruct the device to train the model. The device starts to train the model using its local data once the transmission completed (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 time period. It also checks if it has missed the deadline according to the time needed to finish training and communication. The preceding checking is powered by our IMA dataset described in §3.2.1. The server validates updates based on the checking results 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.

3.4 Experimental Settings

Algorithms. We briefly introduce the algorithms explored in our study and leave more details and their hyper-parameters in §4. The algorithms can be divided into three categories: (1) The Basic algorithm, i.e., *FedAvg*, which has been deployed to real systems [5] and is widely used in FL literature [20, 24, 29, 34]. (2) Aggregation algorithms that determine how to aggregate the weights/gradients uploaded from multiple devices, including *q-FedAvg* [29] and *FedProx* [28]. (3) Compression algorithms, including *Structured Updates* [24], *Gradient Dropping* (*GDrop*) [1], and *SignSGD* [4], which compress local models' weights/gradients to reduce the communication cost between devices and the central server.

Metrics. In our experiments, we quantify the impacts 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. Noting that the training time reported by our simulation platform is the running time after the FL system is deployed in real world instead of the time to run simulation on the pure cloud. (3) *Compression ratio*, which is defined as the fraction of the size of compressed gradients to the original size [46]. (4) *Variance of accuracy*, which is calculated as the standard deviation of accuracy across all the devices in the benchmark dataset. This metric indicates the cross-device fairness of an algorithm. Table 2 summarizes the algorithms and their corresponding metrics that we measure.

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

Table 2: Three categories of FL algorithms we choose and their corresponding metrics we measure.

Computing Environment. All experiments are performed on a high-performance computing cluster with Red Hat Enterprise Linux Server release 7.3 (Maipo). The cluster has 10 GPU workers. Each worker is equipped with 2 Intel Xeon E5-2643 V4 processor, 256G main memory, and 2 NVIDIA Tesla P100 graphics cards. In total, the reported experiments cost more than 5,700 GPU-hours.

4 RESULTS

In this section, we report the results on how heterogeneity impacts the performance of the basic *FedAvg* algorithm (§4.1) and advanced FL algorithms proposed by recent FL-related studies (§4.2).

4.1 Impacts on Basic Algorithm's Performance

We first measure the impacts 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 perform the measurement under different numbers of local training epochs, i.e., different numbers of times that the devices use their local data to update the weights of their local models (refer to §2). The number of local training epoch is an important hyper-parameter of *FedAvg* used to balance the communication cost between the server and the devices [20, 28, 30]. We follow previous work [30] to set this number (denoted as *E*) to 1, 5, and 20. Also, we use the learning rate and the batch size recommended by *Leaf* [8] for each ML model. Figure 4 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.

- **Heterogeneity causes non-trivial accuracy drop in FL.** Under heterogeneity-aware settings, the accuracy drops on each dataset across various local training epoch. Specifically, the accuracy drops by an average of 2.3%, 0.5%, and 4% on the existing Femnist, Celeba, and Reddit datasets, respectively. On our M-Type dataset, the accuracy drop is more significant, with an average of 9.2%.
- **Heterogeneity obviously slows down the training process of FL in terms of both training time and training rounds.** We

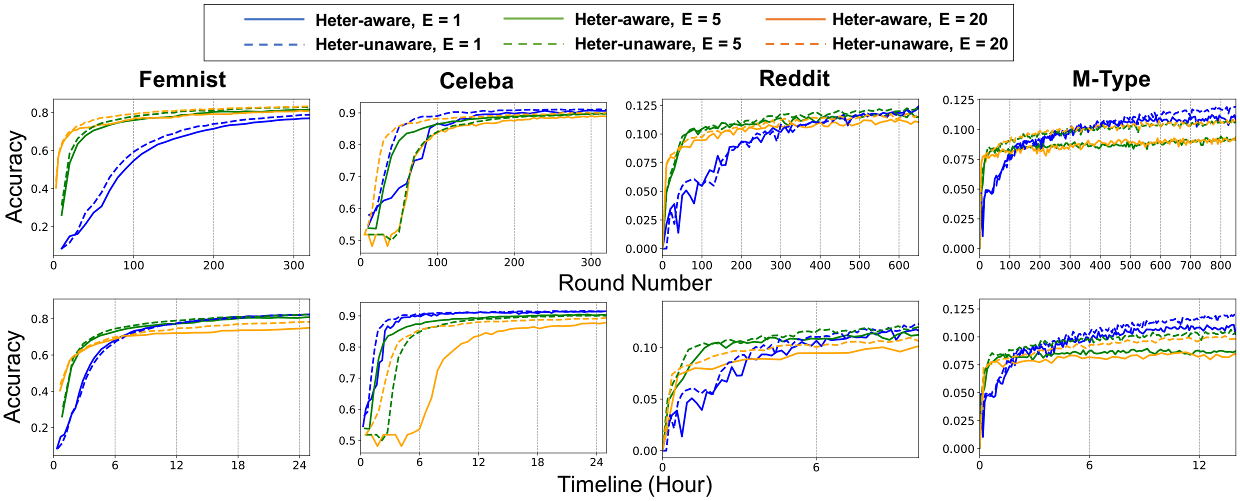

Figure 4: The testing accuracy over time, across different numbers of local training epochs (denoted as *E*)

Dataset	Heter.	Algo.	Average	Worst 10%	Best 10%	Var. ×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 of accuracy across devices.

first analyze the results in terms of training time. Under each setting of the local training epoch, the 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 $E = 20$), with an average of 1.74×. In addition, we find that the 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 analyze 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 $E = 20$), with an average of 1.42×.

4.2 Impacts on Advanced Algorithms' Performance

We now measure the impacts of heterogeneity on advanced FL algorithms, e.g., model aggregation and gradient compression.

4.2.1 ***Aggregation Algorithms.*** The aggregation algorithm is a key component in FL that determines how to aggregate the weights or gradients uploaded from multiple devices. Besides *FedAvg*, various aggregation algorithms are proposed to improve efficiency [28, 37, 38], ensure fairness [29], preserve privacy [6, 38], etc. To study how heterogeneity affects the performance of aggregation algorithms, we focus on two representative ones: *q-FedAvg* [29] and *FedProx* [28], 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 with 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. For *q-FedAvg*, the results are shown in Table 3, which illustrates the the same metrics as evaluated by *q-FedAvg* (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). For *FedProx*, the results are shown in Figure 5, which presents the accuracy changes by round. Due to space limit, we show only the results on two datasets, i.e., one dataset using the CNN model (Femnist) and another dataset using the LSTM model (M-Type). Our observations are as follows.

• ***q-FedAvg*** **that is supposed to address fairness issues is less effective in ensuring fairness under heterogeneity-aware settings.** According to Table 3, under heterogeneity-unaware settings, the worst 10% accuracy of *q-FedAvg* is higher than that of *FedAvg* and *q-FedAvg* also obtains lower variance of accuracy 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. It is probably because *q-FedAvg* cannot tackle the bias in device selection introduced by state heterogeneity (see details in §5.3), which makes *q-FedAvg* less effective in ensuring fairness.

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 gradients compression algorithms. Numbers in the 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 compressed gradients to the original size.

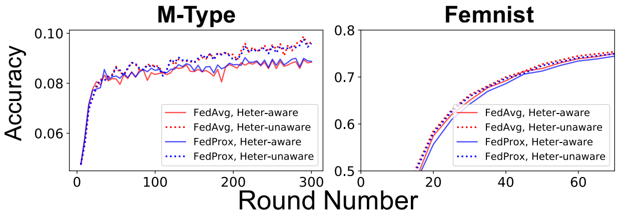

Figure 5: The training performance of *FedProx* and *FedAvg* with and without heterogeneity.

• ***FedProx* is less effective in improving the training process with heterogeneity considered.** According to Figure 5, on M-Type, *FedProx* only slightly outperforms *FedAvg*, and the heterogeneity causes an accuracy drop of 7.5%. On Femnist, *FedProx* achieves the same performance as *FedAvg* under heterogeneity-unaware settings and slightly underperforms *FedAvg* under heterogeneity-aware settings. The heterogeneity causes an accuracy drop of 1.2%. Note that *FedProx* incorporates hardware heterogeneity into its design while leaving state heterogeneity unsolved. We manually check the involved devices and find that only 51.3% devices have attended the training when the model reaches the target accuracy. As a result, the model may have been dominated by these active devices and perform badly on other devices.

4.2.2 ***Gradient Compression Algorithms****.* The cost of device-server communication is often reported as a major bottleneck in FL [21], so we first investigate gradient compression algorithms that are extensively studied to reduce the communication cost. Specifically, we focus on three well-adopted gradient compression algorithms: *Structured Updates* [24], *Gradient Dropping* (*GDrop*) [1], and *SignSGD* [4]. For each of them, we tune the hyper-parameters to achieve the highest accuracy through massive experiments. As a result, for *Structured Updates*, we set the max rank of the decomposited matrix to 100; for *GDrop*, we set the weights 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 use *FedAvg* with no compression as the baseline for comparison. Besides accuracy and training time/rounds, we also use compression ratio (described in §3.4) as the measurement metrics of these algorithms. We present the metric values of the three compression algorithms as well as the baseline under heterogeneity-unaware and heterogeneity-aware settings in Table 4. Similar to §4.2.1, we report only the results on Femnist and M-Type. We summarize our findings as follows.

• **Heterogeneity introduces a similar accuracy drop to compression algorithms as it does to the basic algorithm.** We measure 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 the one (3.2% on average) that we observe in §4.1. On average, the accuracy drops by 1.7% on Femnist and 5.3% on M-Type. It is reasonable because heterogeneity will not affect the compressed gradients.

• **Gradient compression algorithms can hardly speed up the model convergence under 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 to 0.93× at most) and lengthened in most cases. For example, on M-Type under heterogeneity-aware environment, 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 the 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, the accuracy increases slowly when the gradients are compressed and the heterogeneity is introduced (refer to §4.1), thus taking more rounds to reach the target accuracy.

5 ANALYSIS OF IMPACT FACTORS

Given the non-trivial negative impacts of heterogeneity shown in the previous section, we dive deeper to analyze the main factors of these impacts. In this section, we focus on *FedAvg*, considering its wide usage in practical applications. Specifically, we first break down heterogeneity into two types, i.e., state heterogeneity and hardware heterogeneity, to analyze their individual impacts (§5.1). Then we report two phenomena that are particularly obvious under heterogeneity-aware settings according to our experiments: (1) selected devices can fail to upload their model updates for several reasons, which we call *device failure* (§5.2); (2) devices that succeed in uploading still have biased contribution to the global model, which we call *participant bias* (§5.3).

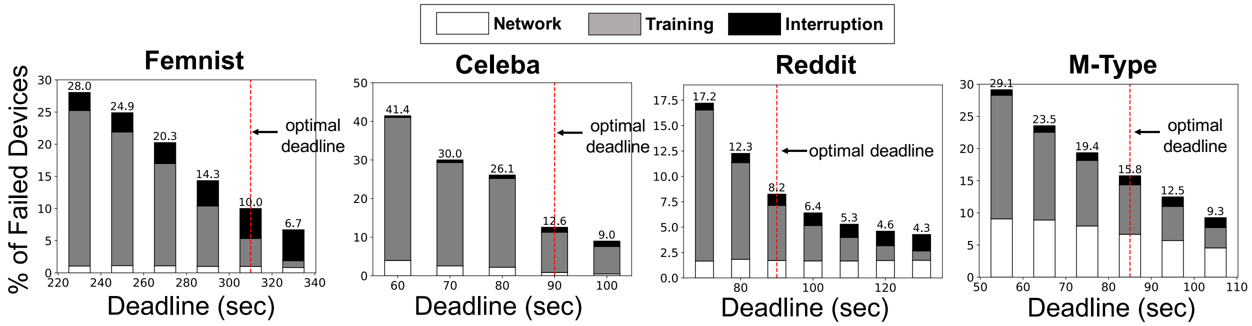

Figure 6: The prevalence of different failure reasons. The optimal deadline (red line) refers to the one that achieves the shortest training time.

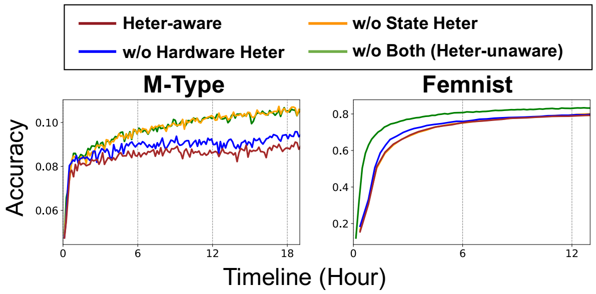

Figure 7: A breakdown of the impacts of different types of heterogeneity. State heterogeneity causes more performance degradation than hardware heterogeneity. "Heter" is short for heterogeneity.

5.1 Breakdown of Heterogeneity

The preceding results indicate the joint impacts from two types of heterogeneity. To analyze their individual impact, we disable the hardware heterogeneity, i.e., all the devices have the same computational and communication capacity (noted as "w/o hardware heter"). Similarly, we disable the state heterogeneity, i.e., devices are always available at any time and will not drop out (noted as "w/o state heter"). We show the accuracy changes with the training time in Figure 7.

• **Both state heterogeneity and hardware heterogeneity slow down the model convergence.** According to Figure 7, state heterogeneity leads to comparable increase of training time to hardware heterogeneity, i.e., 1.72× vs. 1.26× on M-Type and 2.34× vs. 2.62× on Femnist. It is reasonable because both drop-out (introduced by state heterogeneity) and low-end devices (introduced by hardware heterogeneity) affect the training time.

• **State heterogeneity is more influential than hardware heterogeneity on the model accuracy.** As shown in Figure 7, 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 and only a small amount of work [9, 25, 28, 37] explores hardware heterogeneity (refer to §2). Our results show that state heterogeneity is more responsible for the model accuracy drop, which explains why *FedProx* (it considers hardware heterogeneity) is less effective given both types of heterogeneity (refer to §4.2.1).

5.2 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 the model convergence and cause a waste of valuable device resources (computations, energy, etc.). However, device failure is seldom studied in prior work, probably because it is directly related to the FL heterogeneity.

Heuristically, we categorize device failure to three possible causes: (1) ***Network failure*** is detected if the device takes 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 the user interruption, e.g., the device is uncharged during training. (3) ***Training failure*** refers to the case when the device takes too much time on training.

To understand device failure, we zoom into the previous experiments under varied round deadlines. We vary the deadline because we find that the proportion of failed devices is greatly affected by it. Similar to §5.1, we will also check hardware heterogeneity's and state heterogeneity's influence on device failure. The key questions we want to answer here are: (1) how often the devices may fail and what the corresponding reasons for the failure are; (2) and which type of heterogeneity is the major factor. The results are illustrated in Figures 6 and 8, 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 the 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 that: (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 the interruption failure into three sub-categories corresponding to

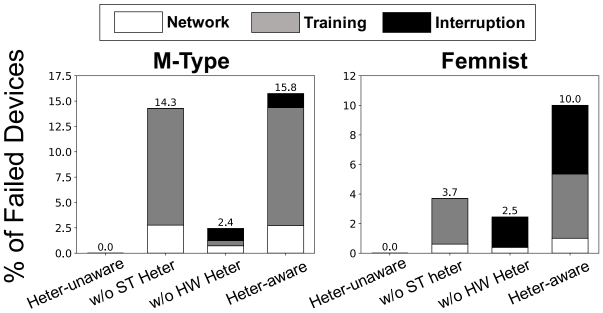

Figure 8: Different kinds of heterogeneity's influence on device failure.

three restrictions on training [5]. Specifically, results show that the training process is interrupted by user interaction, battery charged-off, and network changes with a probability of 46.06%, 36.96%, and 17.78% respectively. (3) Training failure is heavily affected by the deadline. This type of failure accounts for the majority of the device failure when the deadline is set too tight. Even with the optimal deadline setting, this type of failure still occurs because we observe that some low-end devices with too many local data sometimes fail to meet the deadline.

- **Hardware heterogeneity leads to more device failure than state heterogeneity.** According to Figure 8, hardware heterogeneity is more responsible for the device failure. For example, on M-Type, hardware heterogeneity causes 14% failed devices on average while state heterogeneity causes only 2.5%. It is probably because when hardware heterogeneity is considered, there are low-end devices that suffer longer training time.

5.3 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, thus making some devices under-represented. Due to state heterogeneity, devices frequently used by users are less likely to check in. Due to hardware heterogeneity, low-end devices are less likely to upload their updates to the central server.

To measure the participant bias introduced by heterogeneity, we run the same FL tasks in §4.1. We take the amount of computation to reflect the participation degree of different devices. Since it is difficult to compare the computation of different models directly, we divide them by the amount of computation for a training epoch (noted as computation loads). Figure 9 illustrates the distribution of computation loads across devices when the global model reaches the target accuracy. Similar to §5.1, we also break down to explore the impacts of different types of heterogeneity. We summary our findings as follows.

- **The computation loads get more uneven under heterogeneity-aware settings.** The variance is increased by 2.4× (Reddit) to 10.7× (Femnist). Compared to heterogeneity-unaware environment where every device participates with an equal probability, in the heterogeneity-aware environment, the computation loads have a trend of polarization. On Celeba, the maximum computation load increases by 1.17×.

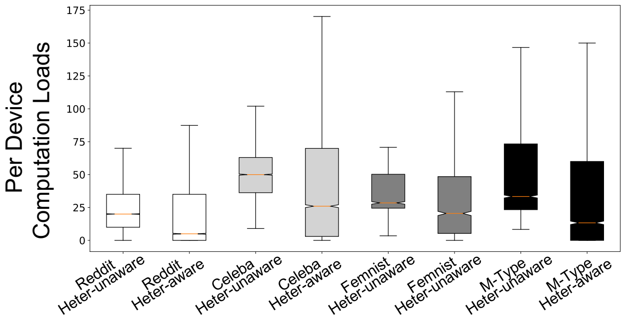

Figure 9: The distribution of computations across devices during FL training.

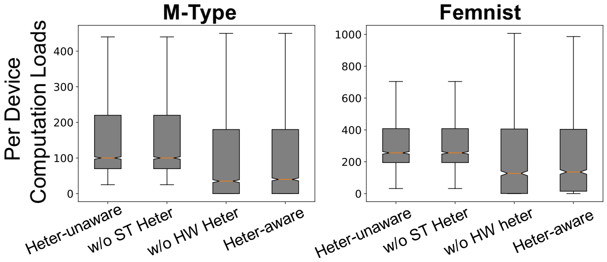

Figure 10: A breakdown of the impacts of different types of Heterogeneity on participant bias.

- **The number of inactive devices increases significantly under heterogeneity-aware settings.** The median computation load drops by 28% (Femnist) to 75% (Reddit), indicating that more inactive devices appear. Compared to the heterogeneity-unaware environment where top 30% of the devices contribute 54% of the total computation, in the heterogeneity-aware environment, top 30% of the devices contribute 81% of the total computation, putting the inactive devices at a disadvantage.

- **Up to 30% devices have not participated in FL process when the global model reaches the target accuracy under heterogeneity-aware settings.** To investigate the reasons for these inactive devices, we inspect the percentage of participating devices over time and demonstrate the result in Figure 11. We find that when the model reaches the target accuracy (6-24 hours in our experiments), more than 30% devices have not participated. In the heterogeneity-unaware environment, the participating devices accumulate quickly and soon cover the total population in 12 hours. While in heterogeneity-aware environment, the accumulation speed gets much slower and it takes much longer time to converge (more than 48 hours).

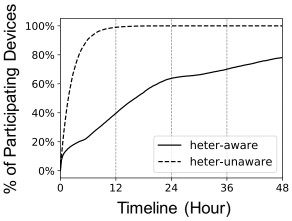

Figure 11: Percentage of participating devices over time.

• **State heterogeneity is more responsible for participant bias.** As shown in Figure 10, state heterogeneity is the main reason for computation bias. It causes the similar computation distribution as the one in heterogeneity-aware environment. It 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 if the device can meet the state criteria (refer to §3.2.1).

6 IMPLICATIONS

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

6.1 For FL Algorithm Designers

Taking heterogeneity into consideration. As demonstrated in our study, heterogeneity introduces non-trivial accuracy drop and training slowdown in FL, as well as affects the effectiveness of some proposed methods. These findings encourage researchers to consider heterogeneity, especially state heterogeneity, when they practice on FL. On the one hand, when designing approaches or algorithms, researchers should consider circumstances that are common in heterogeneity-aware environment but do not exist in heterogeneity-unaware environment. 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, additional system overhead of the algorithm may further widen the gap in training time between different devices, which should be considered during the evaluation.

Reducing device failure by a "proactive alerting" technique. In §5.2, we find that around 10% of devices fail to upload their model updates under typical settings. The reasons include excessive training time, unstable network, and device drop-out caused by state changes. Existing efforts have explored dynamic deadline [26] and tolerating partial work [28] to handle the device failure. However, these algorithms are inadequate to handle the failure caused by unstable network and drop-out because they are highly dependent on the device's states. One may explore a "proactive alerting" technique by predicting the device's future states and network condition based on historical data. The server should assign a low priority to the devices that are likely to drop out. In this way, the overall device failure can be reduced and more updates can be aggregated thus saving the hardware resource and accelerating learning process.

Resolving bias in device selections. In §5.3, we find that the global model is dominated by some active devices (top 30% of devices can 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 the 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. The "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 tuned. What is more, adjusting the local objective (loss function) or re-weighting updates can be possible alternatives.

6.2 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 shall perform in real-world settings. However, existing platforms [8, 18, 39, 42, 44] fail to incorporate heterogeneity into their design. Our work provides a reference implementation and can be easily integrated into these FL platforms. We also encourage system providers to collect their own data that fit different scenarios to further help the FL community.

Optimizing on-device training time, instead of optimizing compression in unmetered (e.g., WiFi) networks. In §4.2.2, we find that gradient compression algorithms can hardly speed up model convergence. The time spent on communication is relatively small in the WiFi environment, compared to the time spent in 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) [16, 48] and using hardware AI accelerators like mobile GPU and digital signal processor (DSP).

7 DISCUSSION

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

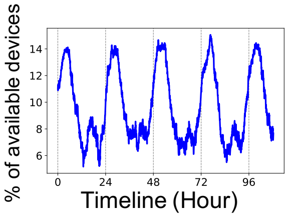

Figure 12: Percentage of available devices over time.

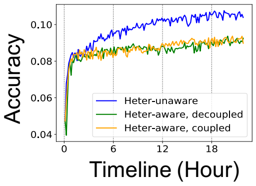

Figure 13: Decoupling is verified on M-Type.

Bias of our IMA dataset. The device state traces (§3.2.1) are collected from our IMA (app-specific) whose users mainly reside in Southeast Asia and Latin America (geo-specific). The traces may not be fully representative to other FL scenarios. However, we believe that our findings are still faithful because (1) FL task is always app-specific and improving IMA experience is a key scenario of FL [5, 17, 53]; (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 [53] as aforementioned. 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. Similar to all existing FL platforms [8, 18, 39, 42, 44], our platform (§3.3) performs FL tasks in a simulation way. We carefully design the platform to simulate the real-world FL systems by considering the 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 12) and failure proportion (Figure 6), are consistent with the results reported from a large-scale FL deployment by Google [5]. Therefore, we believe that our findings are still valid.

Validity of randomly assigning state traces and training data to devices. In practice, the heterogeneity is inherently coupled with the non-IID data distribution [21]. In this study, we decouple the heterogeneity from the 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 13, the gap between the coupled case and the decoupled case is trivial compared to the gap between the heterogeneity-unaware and heterogeneity-aware settings. It justifies our design to decouple heterogeneity from any third-party datasets.

Other types of heterogeneity. In this paper, we focus on the impacts of hardware and state heterogeneity. In fact, there also exist other types of heterogeneity in FL. One is data heterogeneity [21, 27] that 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 [9, 23, 30] 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 [21], like heterogeneity on software or platform, are highly relevant to the implementation of an FL system and hard to generalize. We plan to leave them for future work.

8 CONCLUSION

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

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China under the grant number 2020YFB2104100, the National Natural Science Foundation of China under grant numbers 61725201, J1924032, and 62032003, the Beijing Outstanding Young Scientist Program under the grant number BJJWZYJH 01201910001004, and the Alibaba-PKU Joint Research Program. Mengwei Xu was supported by the NSFC under the grant number 61921003.

REFERENCES

[1] Alham Fikri Aji and Kenneth Heafield. 2017. 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*, Martha Palmer, Rebecca Hwa, and Sebastian Riedel (Eds.). 440–445.

[2] Aaron Archer, Silvio Lattanzi, Peter Likarish, and Sergei Vassilvitskii. 2017. Indexing public-private graphs. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*. 1461–1470.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459* (2018).

[4] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. 2019. 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.*

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: system design. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019.*

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

[7] Keith Bonawitz, Fariborz Salehi, Jakub Konečnỳ, Brendan McMahan, and Marco Gruteser. 2019. Federated learning with autotuned communication-efficient secure aggregation. *arXiv preprint arXiv:1912.00131* (2019).

[8] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: a benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[9] Zheng Chai, Hannan Fayyaz, Zeshan Fayyaz, Ali Anwar, Yi Zhou, Nathalie Baracaldo, Heiko Ludwig, and Yue Cheng. 2019. Towards taming the resource and data heterogeneity in federated learning. In *Proceedings of 2019 USENIX Conference on Operational Machine Learning (OpML 19)*. 19–21.

[10] Yang Chen, Xiaoyan Sun, and Yaochu Jin. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems* (2019).

[11] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. 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.* 750–762.

[12] Zhenpeng Chen, Xuan Lu, Wei Ai, Huoran Li, Qiaozhu Mei, and Xuanzhe Liu. 2018. Through a gender lens: learning usage patterns of emojis from large-scale Android users. In *Proceedings of the 2018 World Wide Web Conference, WWW 2018.* 763–772.

[13] Zhenpeng Chen, Huihan Yao, Yiling Lou, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, and Xuanzhe Liu. 2021. An empirical study on deployment faults of deep learning based mobile applications. In *Proceedings of the 43rd International Conference on Software Engineering, ICSE 2021.* Accepted to appear.

[14] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: extending MNIST to handwritten letters. In *Proceedings of 2017 International Joint Conference on Neural Networks (IJCNN).* 2921–2926.

[15] Eclipse. 2020. Deep Learning for Java. https://deeplearning4j.org/. Accessed Mar 16, 2020.

[16] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: a survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.

[17] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[18] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, et al. 2020. Fedml: a research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).

[19] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. 2018. Ai benchmark: running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV).* 288–314.

[20] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488* (2019).

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

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

[23] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[24] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[25] Yassine Laguel, Krishna Pillutla, Jérôme Malick, and Zaid Harchaoui. 2020. Device heterogeneity in federated learning: a superquantile approach. *arXiv preprint arXiv:2002.11223* (2020).

[26] Li Li, Haoyi Xiong, Zhishan Guo, Jun Wang, and Cheng-Zhong Xu. 2019. SmartPC: hierarchical pace control in real-time federated learning system. In *Proceedings of 2019 IEEE Real-Time Systems Symposium (RTSS).* 406–418.

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

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020.*

[29] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair resource allocation in federated learning. In *Proceedings of 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017.* 1273–1282.

[31] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).

[32] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of 2019 IEEE Symposium on Security and Privacy (SP).* 691–706.

[33] Christian Meurisch, Bekir Bayrak, and Max Mühlhäuser. 2020. Privacy-preserving AI services through data decentralization. In *Proceedings of the Web Conference 2020, WWW 2020.* 190–200.

[34] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. 2019. Agnostic federated learning. *arXiv preprint arXiv:1902.00146* (2019).

[35] The Chinese University of Hong Kong Multimedia Laboratory. 2020. Large-scale CelebFaces Attributes (CelebA) Dataset. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. Accessed May 22, 2020.

[36] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. 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).

[37] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proceedings of ICC 2019-2019 IEEE International Conference on Communications (ICC).* IEEE, 1–7.

[38] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2019. Secure federated submodel learning. *arXiv preprint arXiv:1911.02254* (2019).

[39] PaddlePaddle. 2020. Federated deep learning in PaddlePaddle. https://github.com/PaddlePaddle/PaddleFL. Accessed Jan 28, 2020.

[40] PushShift.io. 2020. Reddit Dataset. https://files.pushshift.io/reddit/. Accessed May 22, 2020.

[41] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2019. Fedpaq: a communication-efficient federated learning method with periodic averaging and quantization. *arXiv preprint arXiv:1909.13014* (2019).

[42] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017* (2018).

[43] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. In *Proceedings of Advances in Neural Information Processing Systems.* 4424–4434.

[44] Tensorflow. 2020. TensorFlow Federated: Machine Learning on Decentralized Data. https://www.tensorflow.org/federated. Accessed Jan 28, 2020.

[45] Wikipedia. 2020. California Consumer Privacy Act. https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act. Accessed Feb 5, 2020.

[46] Wikipedia. 2020. Compression ratio. https://en.wikipedia.org/wiki/Compression_ratio. Accessed Oct 18, 2020.

[47] Wikipedia. 2020. General Data Protection Regulation. https://en.wikipedia.org/wiki/General_Data_Protection_Regulation. Accessed Feb 5, 2020.

[48] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. 2019. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392* (2019).

[49] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. 2019. A first look at deep learning apps on smartphones. In *Proceedings of The World Wide Web Conference.* 2125–2136.

[50] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. 2018. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–26.

[51] Mengwei Xu, Feng Qian, Mengze Zhu, Feifan Huang, Saumay Pushp, and Xuanzhe Liu. 2019. Deepwear: Adaptive local offloading for on-wearable deep learning. *IEEE Transactions on Mobile Computing* 19, 2 (2019), 314–330.

[52] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. DeepCache: principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking.* 129–144.

[53] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[54] Saxena Yogesh, Shrivastava Abha, and Singh Priyanka. 2014. Short communication mobile usage and sleep patterns among medical students. *Indian J Physiol Pharmacol* 58, 1 (2014), 100–103.

[55] Jinliang Yuan, Mengwei Xu, Xiao Ma, Ao Zhou, Xuanzhe Liu, and Shangguang Wang. 2020. Hierarchical Federated Learning through LAN-WAN Orchestration. *arXiv preprint arXiv:2010.11612* (2020).