Empirical Analysis of Federated Learning in Heterogeneous Environments

Ahmed M. Abdelmoniem*
Queen Mary University of London
United Kingdom

Chen-Yu Ho
KAUST
Saudi Arabia

Pantelis Papageorgiou†
KAUST
Saudi Arabia

Marco Canini
KAUST
Saudi Arabia

Abstract

Federated learning (FL) is becoming a popular paradigm for collaborative learning over distributed, private datasets owned by non-trusting entities. FL has seen successful deployment in production environments, and it has been adopted in services such as virtual keyboards, auto-completion, item recommendation, and several IoT applications. However, FL comes with the challenge of performing training over largely heterogeneous datasets, devices, and networks that are out of the control of the centralized FL server. Motivated by this inherent setting, we make a first step towards characterizing the impact of device and behavioral heterogeneity on the trained model. We conduct an extensive empirical study spanning close to 1.5K unique configurations on five popular FL benchmarks. Our analysis shows that these sources of heterogeneity have a major impact on both model performance and fairness, thus shedding light on the importance of considering heterogeneity in FL system design.

Keywords: Federated Learning, Heterogeneity, Performance, Fairness

ACM Reference Format:
Ahmed M. Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, and Marco Canini. 2022. Empirical Analysis of Federated Learning in Heterogeneous Environments. In *2nd European Workshop on Machine Learning and Systems (EuroMLSys '22), April 5–8, 2022, RENNES, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3517207.3526969

*Corresponding author. Work primarily done while at KAUST.
†Work done during an internship at KAUST.

EuroMLSys '22, April 5–8, 2022, RENNES, France

ACM ISBN 978-1-4503-9254-9/22/04…$15.00
https://doi.org/10.1145/3517207.3526969

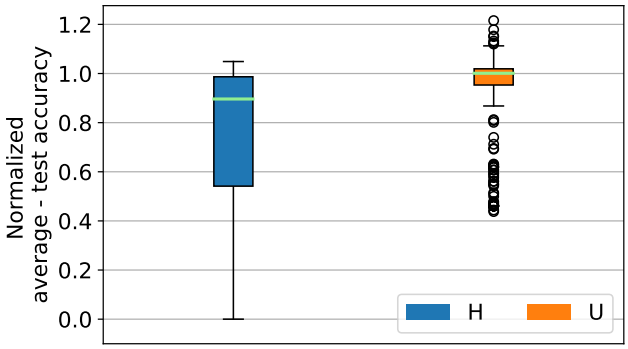

Figure 1. Impact of heterogeneity on model performance. The test accuracy is averaged over learners and normalized by the baseline accuracy of the benchmark in default setting.

1 Introduction

The growing computational power of end-user devices (e.g., mobile phones) coupled with the availability of rich sets of distributed data and concerns over transmitting private information make offloading model training to these devices increasingly attractive. Federated Learning (FL) [32] is a ML approach wherein client devices owning different sets of data collaborate with the assistance of a central FL server to learn a global model without ever transmitting the private data. Recently, FL has gained popularity as a range of practical applications and systems are readying deployment [7, 26, 53, 54]. However, a major challenge with FL, is dealing with the intrinsic heterogeneity of real-world environments.

To appreciate the extent of the problem, Figure 1 contrasts two scenarios: (**U**) the ideal case where devices have uniform computational and network access capabilities and are sampled uniformly at random; and (**H**) the realistic case where client devices have heterogeneous hardware and network resources and their availability varies over time. The box-plot presents the test accuracy across a large number of experiments sweeping several hyper-parameters for five FL benchmarks (c.f. §4 for details). The figure shows that heterogeneity has a significant impact on model performance: for 50% of the experiments, the average accuracy is below

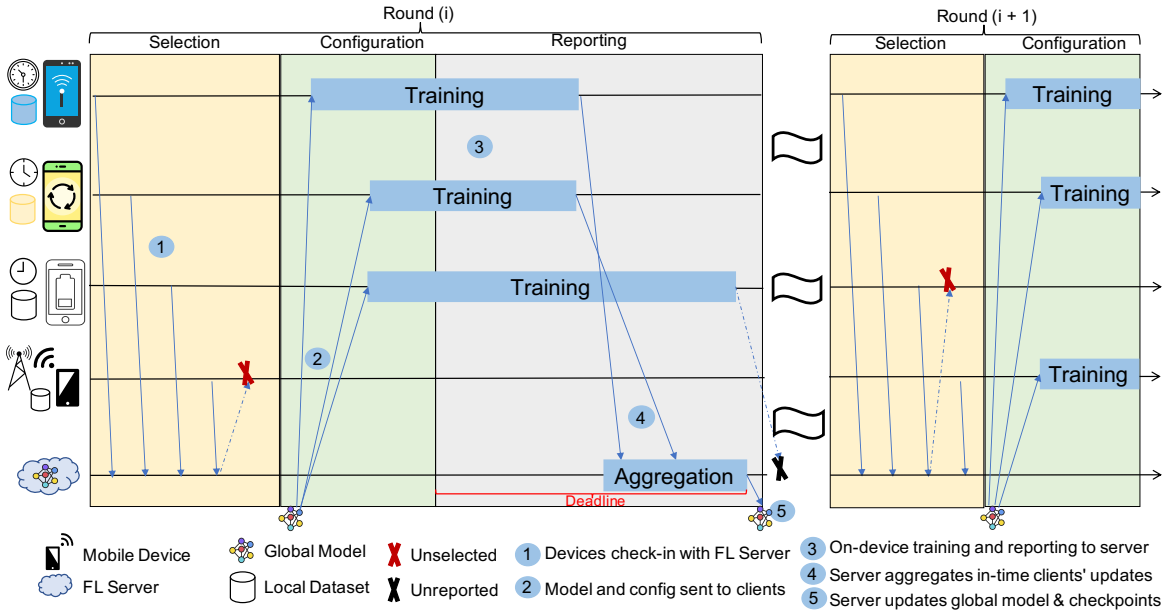

Figure 2. Phases of the typical federated learning process.

0.88× relative to corresponding baselines, and at the extreme, training does not converge.

In this work, we focus on Device heterogeneity (**DH**), which is caused by variations in client devices' computation and communication speeds and Behavioral heterogeneity (**BH**), which is caused by the churn patterns of the client devices. Despite a growing body of work in FL (see [22] for a detailed survey), we find that a systematic assessment of the impact of heterogeneity on the performance and fairness of FL trained models has received little attention. Existing works [27, 28, 37, 44, 49] primarily focus on methods to mitigate the impact from heterogeneity. In particular, we aim to answer the following questions:

1. How does heterogeneity impact the model performance and introduce bias in the FL training process?
2. How sensitive is the FL training process to the hyper-parameters of both the FL server and the local learning algorithm?

To this end, we conduct an extensive set of experiments to collect measurements for analyzing empirically the impact of heterogeneity on model performance and fairness. Our experiments span $\approx 1.5K$ different configurations for five FL benchmarks resulting in a total of $\approx 5K$ experiments and total compute time of $\approx 36K$ GPU hours.

The experimental results demonstrate that:

- the impact of heterogeneity varies significantly, and, in many cases, it can lead training to divergence.
- both **DH** and **BH** have an impact, and confounded together (**H**) yields even greater impact on both the model's performance and fairness.
- **DH**, **BH**, number of epochs, learning rate are the most important factors contributing to the impact for most benchmarks.

2 Background

We review FL with emphasis on its system design and then cover the impact of various sources of heterogeneity.

As depicted in Fig. 2, training of the global model occurs over a series of rounds, until the model converges to a satisfactory accuracy. At the start of each round, the server waits for available devices to check-in. The server selects a subset of these devices which meet certain conditions such as being idle, and connected to WIFI and a power source. Then, the server sends the current version of the global model along with the necessary configurations (i.e., hyper-parameter settings) to the selected participant. Each learner performs an equal number of local optimization steps as controlled by the *Local Epochs* hyper-parameter set at the server. Then, learners send their updated models (or a model update, i.e., the delta from the global model) to the server. Typically, all communications are encrypted and some work proposed using differential privacy to prevent information leakage [1]. Finally, the server aggregates, with the global model, the model updates sent by the clients (possibly via secure aggregation [8]) and then checkpoints the new global model.

Beyond maintaining the global model, as shown in Fig. 2, the server performs three phases that allow FL to be practical at scale: selection, configuration and reporting management, which are described below [7].

Selection: When each round starts, the server first waits for available devices to check-in. This takes place within the

Selection Time Window. From the set of connected clients, the server samples up to *Selection Count* devices that will perform the training. If there are fewer devices by the end of the time window, the server progresses when at least there are *Min Selection Count* devices; else it aborts the round.

Configuration: After sufficient number of clients are selected, the server proceeds with sending the current version of the global model and the hyper-parameters of the training to the selected clients. Then, the clients train the received model on their local datasets using the hyper-parameters configuration pre-set by the server (e.g., number of epochs, batch size, learning rate, optimizer, etc).

Reporting: After pushing the training procedure to the selected devices, the server waits for the devices to push their updates. The server uses a *Reporting Deadline* as the timeout. The round completes successfully if, by the end of the deadline, at least *Target Update Fraction* of devices report to the server; otherwise, the round fails and the received model updates are ignored.

2.1 Effects of heterogeneity in federated learning

Real FL deployments are exposed to a variety of environmental factors, like differences in data samples, device capabilities, quality of network links and availability (see Fig. 2). As a result, heterogeneity is endemic although its ultimate impact is not directly clear.

For instance, the popular FedAvg algorithm [32] assumes homogeneous devices, with equal participation probability in training. In practice, Device Heterogeneity (**DH**) and Behavior Heterogeneity (**BH**) skew the distribution of participating devices. From a system perspective, slower devices and devices with poor connectivity are less likely to meet the reporting deadline (**DH**). To be less intrusive to the owners of devices, FL systems consider devices to be eligible when they are plugged to a power source, are connected to an unmetered network and are otherwise idle – all factors influenced by user behavior (**BH**).

We view our work as a first step towards characterizing the effects of heterogeneity-induced bias. To make our study concrete and tractable, among the above issues, we focus on both performance and fairness aspects of the FL trained model. Ideally, FL should ensure that the model has a fair representation of all user groups, under some definition of fairness. The existence of bias can be revealed by measuring the level of fairness among the participants.

We use Jain's fairness index [19] to measure the level of fairness in a distribution of values. Jain's index is commonly used as a measure of fairness of the attained throughput among TCP flows that compete for scarce network bandwidth. Jain's fairness index F_I is expressed as $F_I = \frac{(\sum_{i=1}^{N} x_i)^2}{N \times \sum_{i=1}^{N} x_i^2}$, where, N is the number of clients and x_i is the per-client performance measure under consideration for fairness evaluation (typically for us, the clients' test accuracy).

3 Methodology

To characterize and quantify the impact of heterogeneity on model performance and fairness, we use an experimental design approach [14] following these driving questions:

1. Is there a definite trade-off between model performance/fairness and heterogeneity-induced bias?
2. To what extent does **BH** versus **DH** affect the bias and how do their individual effects confound?

3.1 Experimental design

The experimental design approach comprises a careful selection of influencing factors chosen to allow an accurate view of the system's response. Repeating runs allow to account for the variance of individual experiments.

Factors: The factors are the heterogeneity vs. uniform scenario and the many hyper-parameters that influence the FL process. We subdivide the latter into three categories: environment, application-specific, and FL system/algorithm. We list them out in Table 2 along with the ranges of used values.

Experiments: The space of possible instantiations of factors is huge, which makes it hard to use a space-filling approach to cover the large experimental space uniformly. To principally cover the space of experiments, we identify a default configuration for each benchmark and we then systematically perform experiments while varying factors, typically one by one as a deviation from the default configuration. However, certain characteristics of any experiment are random: e.g., the assignment of data partitions to devices, the proportion of device types, or the available clients during selection phase. We control for variance by repeating experiments five times using distinct seeds to initialize randomness.

A noteworthy aspect of a benchmark's default configuration is the reporting deadline. We search for the *critical deadline*, that is, an appropriate minimal round duration such that the fraction of successful clients over the rounds is on average above the target update fraction.

Platform: We use Flash [52] to run realistic experiments. Flash simulates runs of FL applications and faithfully models wall-clock execution time while multiplexing execution of many devices onto a single GPU. Flash bins devices into three capability-classes: low-end (LE), moderate (M) and high-end (HE) devices. The computational speed of these groups follow the execution profiles of three real-world devices [52]. Finally, the devices' network access links in terms of upload and download speeds, are randomly chosen from 20 different distributions covering a wide-variety of network conditions observed in practice. Flash also comes with a trace of user behavior collected from a real-world FL application deployed across three countries.

Heterogeneity: Since our objective is to tease out the influence of heterogeneous settings, we consider four scenarios for every experiment. The baseline case (**U**) is the ideal scenario where clients are always available and uniform (i.e.,

Table 1. Summary of the benchmarks used in this work.

Task	ML technique	Model	Dataset	Model Size [bytes]	Total Clients	Selection Count (**sc**)	Reporting Deadline (**ddl**) [*s*]	Maximum Sample Count	Learning Rate	Quality metric Accuracy	Divergence Threshold
Image Classification	CNN	2 Conv2D Layers	FEMNIST [13]	26,414,840	3,400	100	60	340	0.01	79.91%	<17%
		1 Conv2D Layer	CelebA [29]	124,808	9,343	100	15	30	0.01	90.63%	<59%
Next Word Prediction	RNN	LSTM	Reddit [41]	24,722,496	813	100	27	50	0.5	11.88%	<5%
			Shakespeare [32]	3,271,488	1,129	50	142	50	0.8	40.10%	<10%
Cluster Identification	Traditional ML	Logistic Regression	Synthetic [9]	2,400	9,367	50	23	340	0.005	83.00%	<30%

Table 2. FL hyper-parameters. **ddl** and **sc** are the default reporting deadline and selection count as in Table 1.

Hyper-parameter	**Values**
Selection count	[10, 50, 100, 500]
Max samples per client	[100, 200, 300, 400]
Target update fraction	[0.0, 0.3, 0.6, 0.8, 0.85, 0.9]
Deadline	[1.25×**ddl**, 1.5×**ddl**, 1.75×**ddl**, 2×**ddl**]
(LE%, M%, HE%)	[(0.0, 0.0, 1.0), (0.0, 1.0, 0.0), (1.0, 0.0, 0.0), (0.0, 0.5, 0.5), (0.5, 0.5, 0.0), (0.5, 0.0, 0.5), (0.1, 0.1, 0.8), (0.8, 0.1, 0.1), (0.1, 0.8, 0.1), (0.2, 0.2, 0.6), (0.2, 0.6, 0.2), (0.6, 0.2, 0.2), (0.33, 0.34, 0.33)]
Min selection count	[0.25×**sc**, 0.5×**sc**, 0.75×**sc**, **sc**]

their devices are homogeneous in hardware and link speed) and the server sets a large enough deadline for all clients to finish in time. The heterogeneity scenarios are as follows: 1. device heterogeneity (**DH**) - the clients are always online but their device hard and link speed are sampled at random from the real-world trace and link speed distributions, respectively; 2. behavioral heterogeneity (**BH**) - the clients use the moderate device model and same link speed but their availability follows the timeline of the user in the real-world trace [52]; 3. full heterogeneity (**H**) - is the simultaneous combination of (**DH**) and (**BH**) where device model and link speed are sampled at random and client availability follows the real-world user trace.

4 Experimental evaluation

Benchmarks: We use five benchmarks covering a variety of FL applications used in several prior works [27, 28, 32, 52].

Table 1 summarizes the application, dataset, and default configuration of each benchmark.

We partition the training and testing datasets so that each client owns a partition of each dataset as done in prior works. During a training round, a selected client uses samples from its training partition. To evaluate the model accuracy on the testing dataset, we run the test rounds 100 times during the whole experiment (i.e., test rounds are set to run every $\frac{R}{100}$ training round where R is the total training rounds). For each round, we evaluate the model accuracy at every client or from a random sample of 3,500 clients, if the total client count is higher than this cap.

The default configuration of each benchmark is primarily based on the information from prior works [7, 9, 16]. The reporting deadline is set by us based on the search of the critical deadline (§3). For all benchmarks, the server uses a selection time window of 20 seconds and the clients use SGD as their local optimizer. We also note that the default *batch size* is 10, *number of epochs* is 1, *min selection count* is 10 and *update fraction* is 0.8 (80%). The default FL aggregation algorithm we use is FedAvg [32] as employed in [7] wherein the clients upload the model updates (i.e., model deltas) to the server for aggregation.

While our evaluation results are for the more realistic conditions with Non Independent, Identically, and Distributed (Non-I.I.D.) data, we also ran each benchmark using an I.I.D. version of the dataset where each data point is equally likely to be sampled and so all clients have the same underlying distribution of data.. Our analysis of both cases (i.e., non-I.I.D. and I.I.D.) support that the results and observations reported are not due to data heterogeneity.

We run experiments on a GPU cluster. We run $\approx 5K$ experiments, requiring a total of $\approx 36K$ GPU-hours (≈ 4 years). Next, we discuss the results in detail. We mainly present the average test accuracy and Jain's fairness index, both normalized by the corresponding values of the baseline (i.e., the setting (**U**) with default configurations). In the figures, the missing values marked as (*), represent the divergent cases.

4.1 Heterogeneous settings and their impact

Fig. 3 presents the model performance (the average test accuracy across devices) and fairness (based on Jain's fairness index) for every benchmark contrasting the heterogeneous scenarios (**BH**, **DH**, **H**) to uniform (**U**). We observe that both **DH** and **BH** have an impact, and generally, they together confound (**H**) to yield even greater impact on both the model's performance and fairness.

The impact of **BH** is more significant for benchmarks that either have a relatively small client population (Reddit) or have a small number of data samples per client (CelebA and Synthetic). These cases are more sensitive to variations in device availability as it affects the quantity and diversity of

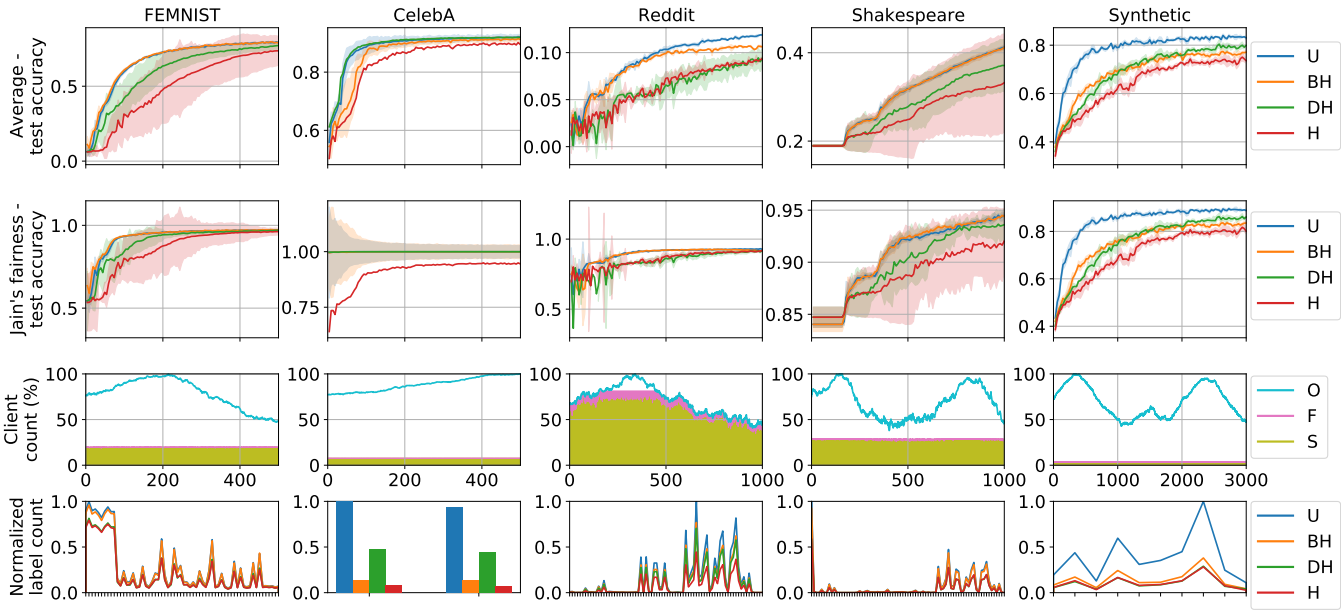

Figure 3. Impact of different heterogeneity settings on model performance and level of fairness for different FL benchmarks. The 1^{st} and 2^{nd} rows show the median value (solid line) and one standard deviation (filled range) among all seeds for the average test accuracy and Jain's fairness index, respectively. The 3^{rd} row shows the online (O), failed (F) and successful (S) client counts throughout rounds. The 4^{th} row shows normalized frequency for each label on which the clients trained throughout training. Note that, CelebA (binary classification) has only 2 labels, thus using grouped bars. X-axis label for rows 1, 2, and 3 is round number and is unique data labels for row 4.

the training data. For the converged runs, the degradation caused by **BH** on average performance and fairness varies from 1.1× to 3.9× and from 1× to 2.2×, respectively.

The impact of **DH** is in general more profound despite all devices are always available. This is because the device type is determined at random following a distribution skewed in favor of low-end devices. Since, the device's ability to finish training before the deadline depends on the time needed to process the device's data samples, low-end devices are more likely to fail to finish the training in time. In our **DH** and **H** experiments, there are on average 55% LE, 44% M, and 1% HE devices. Over the converged runs, the degradation caused by **DH** on average performance and fairness ranges from 1.13× to 4.1× and from 1× to 1.96×, respectively.

The composition of device types and client availability is the main contributor to the average performance and fairness degradation seen for **H**. This degradation can be attributed to the confounding effect of **DH**, which results in a mixture of device types dominated by low-end devices and **BH**, which results in unavailability of moderate and high-end devices. As a result, within heterogeneity, the global model is updated using updates from fewer clients than the uniform case. This is supported by the lower number of unique clients (or data samples) as indicated by label frequency in last row of Fig. 3.

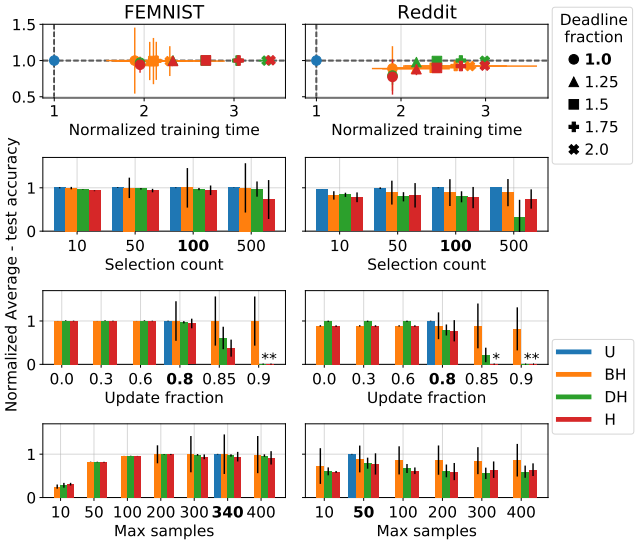

Figure 4. The sensitivity of the average accuracy to the choice of the FL hyper-parameters for FEMNIST and Reddit Benchmarks. We mark the default settings of the uniform scenario in bold in all following figures.

4.2 FL hyper-parameters and heterogeneity settings

We now study the sensitivity of the heterogeneity settings to the choice of FL hyper-parameters. We here focus on the average accuracy results of FEMNIST and Reddit. However, we note that the observations from the other benchmarks are mostly in line with the ones we discuss next.

We observe that the average performance is quite sensitive to the choice of the FL hyper-parameters as shown in Fig. 4. We describe the effects of each of hyper-parameter below.

Reporting Deadline: The deadline is one of the key FL hyper-parameters, which directly influences the success rate of the clients. The default setting is set to allow, on average, just enough fraction of clients to submit the updates in time. The results show that increasing the deadline results in improvement of the obtained average test accuracy. Moreover, the improvements are more pronounced for **DH** and **H** settings. However, we observe that the improvements flatten at a certain point after which the extra time spent in training due to waiting for the deadline would be wasteful. Hence, there is a clear trade-off between the performance and run-time costs which can be mainly controlled by tuning the reporting deadline.

Selection Count: We observe that a higher selection client count leads to almost no improvement in the average model performance for **U** and **BH** settings. However, we observe larger variations, with larger numbers of selection count, as indicated by the error bars for **BH**. In contrast, for **DH** and **H**, increasing the selection count leads to severe degradation in the model performance (esp. for Shakespeare). This can be attributed to both the over-fitting of the model due to larger global batch sizes coupled with the increased number of client failures in **DH** and **H** in which the majority of clients are low-end devices.

Target Update Fraction: We note that, for all heterogeneity settings, the target update fraction of uploading clients directly impacts both the performance and introduces a noticeable trade-off. Specifically, higher targets for update fraction means greater likelihood for round failure (i.e., not meeting the target) which results in a model with low levels of quality (or divergence in many cases especially for **DH** and **H**). In contrast, low update fraction can help with reducing the failed rounds but fewer clients would contribute to the aggregated update potentially increasing the model bias. Therefore, the tuning of this hyper-parameter is quite vital for the convergence of the model.

Maximum Samples: is the maximum number of samples each client is allowed to use when participating in training or testing. This is useful for Non-I.I.D. data to bound each client's contributed samples which would make the learning updates more fair among clients. However, the results show lower values of maximum number of samples can limit the learning process (i.e., slow convergence and hence low model performance) which equally impacts all settings. In general,

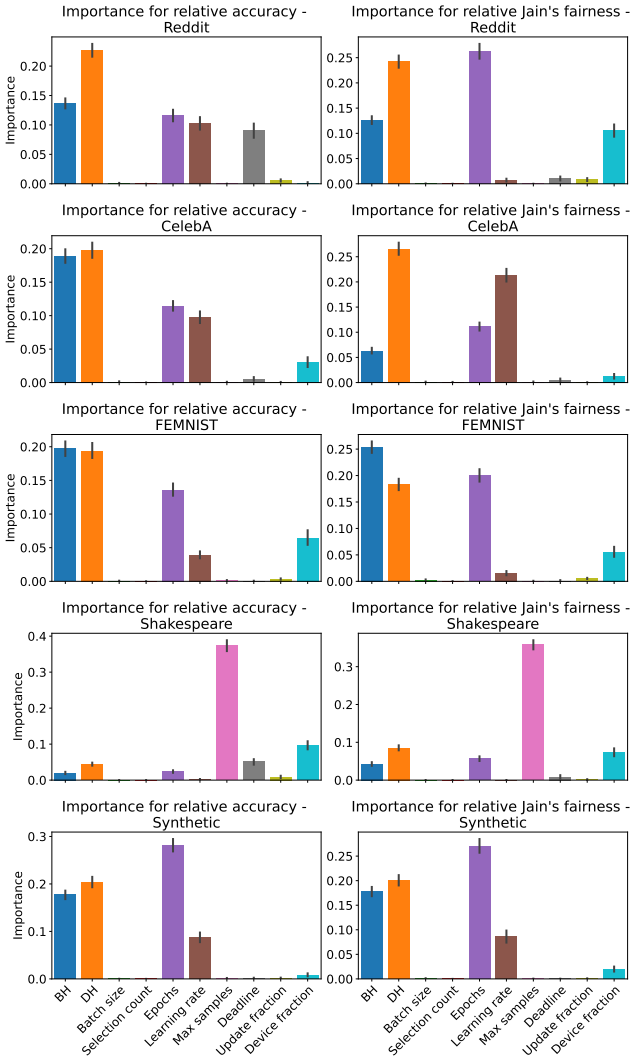

Figure 5. Functional ANOVA analysis.

the results show that, in **BH** settings, the quality of the model are mildly impacted by the choice of the maximum number of samples. In contrast, we observe noticeable impact (and in some cases divergence) for both **DH** and **H** settings. This is because, for large maximum number of samples, clients with low-end devices, needing to process more samples per epoch, will likely fail to report within the deadline. In addition, the maximum number of samples results in a trade-off between ensuring a higher degree of inclusion and diversity of clients' data. This suggests that it should be tuned depending on the distribution of data points available with the clients at the time of training.

4.3 Importance analysis

We now leverage functional ANOVA analysis [18] to evaluate the importance of different factors towards the impact

on model performance. Fig. 5 shows the individual importance of different factors on the experiments. The error bars show the confidence interval around the estimated values based on 100 repeated runs of functional ANOVA. We exclude the divergent runs for this analysis (i.e., runs with average accuracy below the divergence threshold listed in Table 1). We see from the figure that, for most benchmarks, **BH**, **DH**, number of epochs, learning rate are the most important factors. **BH** and **DH** have a noticeable impact on the model performance and fairness, while the remaining hyper-parameters have a comparable or smaller influence to the performance variations. The importance of number of epochs is linked to the default value for the reporting deadline: a larger number of epochs make it more likely to miss the deadline. And, learning rate impacts the ability of achieving reasonable convergence which requires per-task tuning as results suggest that some tasks are more amenable to the learning rate setting than the others.

The exceptions of these observations are with Shakespeare (for both accuracy and fairness performance metrics) where users have many more samples than they can process and increasing the threshold on the maximum samples can cause clients to miss the reporting deadline due to increased computational needs. Therefore, max samples has the highest importance in that case.

4.4 Discussion

We expect that the heterogeneity impact might be reduced by means of proportional load balancing during the selection phase [37], computation offloading techniques at the edge [50], adaptive compression of model for computation [3] and communication [2, 4] or asynchronous mode of model updates [30].

A question also arises: how helpful would it be if one could maintain at the server some level of non-sensitive information about clients' hardware and network characteristics? Then, adaptive methods could be designed for per-client tuning of the hyper-parameters. For instance, to mitigate **BH**, the server may keep historical information on clients' participation and adapt the minimum selection count accordingly. We leave exploration of such strategies for future work.

5 Related Work

Federated Learning (FL): is a paradigm that has recently emerged mainly for privacy-preserving learning. In this paradigm, training is distributed on decentralized devices such as smart edge devices (i.e., mobile or IoT sensor devices) which produces data samples, so that data does not leave the local storage of the data source [25, 32]. For this reason, FL has seen an emerging popularity and is currently deployed for a large number of users to enhance the functionality of the virtual keyboards (e.g., the search suggestion quality[7, 54]). Moreover, to facilitate and expedite research efforts, several works developed FL frameworks for experimentation with FL settings [9, 38, 45, 48, 52]. In this work, we dissect heterogeneity and provide a comprehensive evaluation of their impact on the model quality and fairness.

System heterogeneity: Heterogeneity is one of the major challenges for distributed systems. In datacenters, the compute nodes need to aggregate their local model updates among each other via some form of communication backend (e.g., using parameter server [25, 40] or peer-to-peer collective aggregation [31, 39, 46, 51]). In this context, device heterogeneity results in performance degradation due to stragglers (i.e., slow workers) who slow down the training process [10, 20]. Several works tried to address this problem via system and algorithmic solutions [5, 10, 11, 17, 20, 39, 42]. In FL settings, the heterogeneity is sourced from other system artifacts and is not limited to the heterogeneity in device capabilities. Specifically, data distribution among the clients, client sampling method, and user behavior are other main sources of heterogeneity in FL scenarios.

Improvements in FL: In FL, some proposals try to address the communication bottlenecks during the training via exploiting number of communication reduction techniques like compression, periodic update, and layer-wise asynchronous updates [2, 4, 5, 7, 12, 15, 23, 43, 47, 51]. Other works try to study and improve the privacy guarantees of FL environments [6, 7, 33, 34, 36]. Additionally, others focus on personalizing the global model resulting from FL [21] and minimizing the energy consumption on edge devices [24]. Moreover, recent works highlighted the problems of bias and proposed mitigation schemes to enforce fairer representation of the clients in the trained model [27, 28, 35].

6 Conclusion

We present an experimental study of the impact of heterogeneous learners on the performance and bias in federated learning. We empirically study the factors that play a role in introducing heterogeneity, such as device and behavioral heterogeneity. We evaluate how different FL hyper-parameters amplify the impact of heterogeneity on both model performance and fairness. Our evaluation, shows that heterogeneity can cause up to 4.6× and 2.2× degradation in the average test accuracy and the fairness of the model.

Acknowledgments

We thank Muhammad Bilal for his help with the work.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*.

[2] Ahmed M. Abdelmoniem and Marco Canini. 2021. DC2: Delay-aware Compression Control for Distributed Machine Learning. In *INFOCOM*.

[3] Ahmed M. Abdelmoniem and Marco Canini. 2021. Towards Mitigating Device Heterogeneity in Federated Learning via Adaptive Model Quantization. In *EuroMLSys*.

[4] Ahmed M. Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini. 2021. An Efficient Statistical-based Gradient Compression Technique for Distributed Training Systems. In *MLSys*.

[5] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. 2021. Resource-Efficient Federated Learning. *arXiv 2111.01108* (2021).

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *AISTATS*.

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *MLSys*.

[8] 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 *CCS*.

[9] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. LEAF: A Benchmark for Federated Settings. *arXiv 1812.01097* (2018).

[10] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. 2018. AdaComp : Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *AAAI*.

[11] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. 2016. Revisiting Distributed Synchronous SGD. In *ICLR Workshop Track*.

[12] Yang Chen, Xiaoyan Sun, and Yaochu Jin. 2019. Communication-Efficient Federated Deep Learning With Layerwise Asynchronous Model Update and Temporally Weighted Aggregation. *IEEE TNNLS* (2019).

[13] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *IJCNN*.

[14] Ronald Fisher. 1971. *The Design of Experiments* (9 ed.). Macmillan.

[15] Rishikesh R. Gajjala, Shashwat Banchhor, Ahmed M. Abdelmoniem, Aritra Dutta, Marco Canini, and Panos Kalnis. 2020. Huffman Coding Based Encoding Techniques for Fast Distributed Deep Learning. In *Workshop on Distributed Machine Learning*.

[16] 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:1811.03604* (2018).

[17] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, and Eric P. Xing. 2013. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server. In *NeurIPS*.

[18] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. 2014. An efficient approach for assessing hyperparameter importance. In *ICML*.

[19] Raj Jain, Arjan Durresi, and Gojko Babic. 1999. Throughput fairness index: An explanation. *ATM Forum contribution* 99, 45 (1999).

[20] Jiawei Jiang, Bin Cui, Ce Zhang, and Lele Yu. 2017. Heterogeneity-Aware Distributed Parameter Servers. In *SIGMOD*.

[21] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv 1909.12488* (2019).

[22] 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:1912.04977* (2019).

[23] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. In *Workshop on Private Multi-Party Machine Learning - NeurIPS*.

[24] Li Li, Haoyi Xiong, Zhishan Guo, Jun Wang, and Cheng-Zhong Xu. 2019. SmartPC: Hierarchical Pace Control in Real-Time Federated Learning System. In *RTSS*.

[25] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*.

[26] 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).

[27] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys*.

[28] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair Resource Allocation in Federated Learning. In *ICLR*.

[29] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *ICCV*.

[30] Xiaofeng Lu, Yuying Liao, Pietro Lio, and Pan Hui. 2020. Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing. *IEEE Access* 8 (2020).

[31] Liang Luo, Peter West, Arvind Krishnamurthy, Luis Ceze, and Jacob Nelson. 2020. PLink: Discovering and Exploiting Datacenter Network Locality for Efficient Cloud-based Distributed Training. In *MLSys*.

[32] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[33] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR*.

[34] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy (SP)*.

[35] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. 2019. Agnostic Federated Learning. In *ICML*.

[36] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy (SP)*.

[37] Takayuki Nishio and Ryo Yonetani. 2019. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In *ICC*.

[38] PaddlePaddle.org. 2020. PArallel Distributed Deep LEarning: Machine Learning Framework from Industrial Practice. (2020). https://github.com/PaddlePaddle/PaddleFL

[39] Pitch Patarasuk and Xin Yuan. 2009. Bandwidth optimal all-reduce algorithms for clusters of workstations. *J. Parallel and Distrib. Comput.* 69, 2 (2009), 117 – 124. https://doi.org/10.1016/j.jpdc.2008.09.002

[40] Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. 2019. A Generic Communication Scheduler for Distributed DNN Training Acceleration. In *SOSP*.

[41] Pushshift.io. 2020. Reddit Datasets. (2020). https://files.pushshift.io/reddit/.

[42] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *NeurIPS*.

[43] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2020. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. In *AISTATS*.

[44] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. 2021. Towards Flexible Device Participation in Federated Learning . In *AISTATS*.

[45] 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*

1811.04017 (2018).

[46] Alexander Sergeev and Mike Del Balso. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv:1802.05799* (2018).

[47] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated Multi-Task Learning. In *NeurIPS*.

[48] tensorflow.org. 2020. TensorFlow Federated: Machine Learning on Decentralized Data. (2020). https://www.tensorflow.org/federated

[49] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *NeurIPS*.

[50] Wentai Wu, Ligang He, Weiwei Lin, and Rui Mao. 2020. Accelerating Federated Learning over Reliability-Agnostic Clients in Mobile Edge Computing Systems. *IEEE Transactions on Parallel and Distributed Systems* (2020).

[51] Hang Xu, Chen-Yu Ho, Ahmed M. Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. 2021. GRACE: A Compressed Communication Framework for Distributed Machine Learning. In *ICDCS*.

[52] 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 *The Web Conference*.

[53] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019).

[54] 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 1812.02903* (2018).