Improving the Accuracy of Federated Learning Simulations

Using Traces from Real-world Deployments to Enhance the Realism of Simulation Environments

Alexander Nygård[1]

Supervisor(s): Jérémie Decouchant[1]**, Bart Cox**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Alexander Nygård
Final project course: CSE3000 Research Project
Thesis committee: Jérémie Decouchant, Bart Cox, Qing Wang

Abstract

Federated learning (FL) is a machine learning paradigm where private datasets are distributed among decentralized client devices and model updates are communicated and aggregated to train a shared global model. While providing privacy and scalability benefits, FL systems also face challenges such as client and data heterogeneity, where the training resources and different datasets are non-Independent and Identically Distributed (non-IID). When testing novel FL algorithms or configurations, simulators are used to create controlled environments without needing costly deployments. However, it is important to understand how representative FL simulators are of real-world deployments, and what steps can be taken to bridge the gap between theoretical results and practical implementations. In this paper, we investigate the effects of incorporating traces from a pseudo-real heterogeneous FL deployment in simulated environments. We compare four non-IID attributes, including batch sizes, local epochs, data volume, and data labels, to determine the most influential factors for reproducing the deployment results in simulations. We show that there is an inherent difference between deployments and simulations, despite incorporating identical non-IID conditions. Furthermore, we show that including non-IID data labels in simulations has the most significant impact on recreating the deployment outcome. We also demonstrate that incorporating the other mentioned factors has negligible impact, resulting in similar training performance compared to fully IID simulations. Our results are derived from a 20-client single-server synchronous FL configuration, and additional research is necessary to confirm our findings for larger-scale systems.

1 Introduction

Federated learning (FL) is a machine learning technique that enables multiple decentralized devices to collaboratively learn a shared prediction model while keeping all training data locally [18]. Formalized by a team of Google researchers in 2016 [22], FL addresses significant challenges related to data privacy and security by allowing data to remain on users' devices while training a shared model. This approach is useful in scenarios where data privacy is critical, such as in healthcare or financial services [16]. Despite its benefits, developing FL systems can require more effort than centralized machine learning, due to challenges such as client and data heterogeneity [26].

To overcome these limitations, simulations provide a powerful tool in the development and refinement of FL systems. FL simulation software offers a controlled environment where researchers can experiment with different network configurations and algorithms without the need for costly real-world deployments [24]. For example, simulations help in understanding how novel algorithms and client selection methods affect model performance [2; 13]. This controlled, cost-effective, and scalable testing method plays an important role in advancing FL research and development.

However, the effectiveness of simulations in FL is inherently tied to their fidelity, realism, and scalability [25]. Ensuring that simulations accurately reflect real-world conditions is crucial, as inaccuracies can lead to flawed predictions about the system's actual performance [3]. Challenges include replicating the complex network dynamics, diverse device capabilities, and varied data distributions that characterize real-world deployments [5]. As the scale of FL applications grows, so does the difficulty in simulating these environments effectively.

Contribution In this paper, we explore how the realism of FL simulations can be significantly enhanced by incorporating system traces from deployments. By utilizing detailed system traces, we aim to replicate the nuanced and dynamic behaviors of deployment environments within our simulations, thereby bridging the gap between theoretical performance and practical application. Our approach involves integrating factors such as client and data distribution heterogeneity into simulation environments. Through rigorous experimentation, we demonstrate that simulations augmented with system traces provide a more accurate and reliable prediction of FL performance. This not only aids in the design and optimization of FL algorithms but also reduces the risk of unforeseen issues during actual deployment, ultimately contributing to the robustness and scalability of FL systems in practical scenarios. Specifically, we address the research question: *"How can traces in federated learning deployments be used to improve the accuracy of simulations, and what factors have the most significant impact?"*

To address this question, the paper is structured as follows. In section 2, we provide an overview of FL systems and discuss the importance of simulations in their development. Section 3 presents related work on using traces in FL systems and the effect of non-IID factors in simulations. In section 4, we describe our approach to designing and running the experiments. Section 5 presents the results of our experiments, followed by a discussion of limitations and future work in section 6. In section 7 we highlight the ethical considerations and responsible research practices involved in our study. Finally, we conclude the paper in section 8 by summarizing our findings and highlighting the contributions in this field.

2 Background

2.1 Federated Learning

Since its introduction in 2016 [22], FL has become a popular machine learning technique for decentralized learning. FL systems provide a solution to several centralized machine learning issues, such as privacy concerns and scalability issues [30]. In practice, these benefits are achieved by training a shared model across multiple decentralized devices, each with its local dataset, and aggregating the model updates without sharing the raw data [11].

There exist many configurations of FL systems. These can be categorized based on various factors, such as the feature set

of the data, the topology of the system, and the aggregation method used.

Feature set: The feature set of the clients' data can be categorized as either horizontal or vertical [16]. In horizontal FL, all clients have different datasets, but with the same feature space. In contrast, vertical FL involves clients having different datasets with different feature spaces. Vertical FL therefore aims to combine features from multiple clients and leverage complementary information to train a shared model.

Topology: FL systems can be categorized based on their topology and can be deployed with a single-server, multi-server, or fully decentralized topology [27]. Single-server FL systems have a central server that coordinates the training process, while multi-server systems distribute the training coordination process across multiple servers, often in a hierarchical structure. Fully decentralized systems have no central server, and the clients communicate directly with each other.

Aggregation Method: After individual models are trained by the clients, they must be aggregated to create a shared global model. The aggregation method used in FL systems can vary, with several algorithms available. Notable examples include FedAvg [22], FedProx [19], and FedDyn [1].

Despite the privacy and scalability advantages, FL systems face several challenges, many of which have been heavily examined in recent literature [28; 5; 26; 14]. Frequently discussed challenges include security concerns, privacy considerations, and system heterogeneity. While each influences FL systems in different ways, we have chosen to focus on the non-Independent and Identically Distributed (non-IID) nature of clients and data in this paper.

Client Heterogeneity: Clients in an FL system can have vastly different computational capabilities, memory limitations, energy resources, and storage capacities [14]. Some clients may be powerful servers, while others might be resource-constrained mobile devices or Internet of Things (IoT) sensors. Moreover, the client-side training can be designed to run at times when the device is idle or unused. Varying hardware capabilities, combined with fluctuating device usage, result in differences in how much computation each client contributes to the training process.

Data Heterogeneity: Data heterogeneity refers to the non-IID nature of data across different clients [12]. In FL, clients often collect data in diverse environments, leading to variations in data distributions. In practice, the data often varies in volume and label distribution between clients, partly because clients generate data at different rates [18]. As a result, the global model can converge more slowly and perform poorly on some clients' data [20].

There exist many additional factors influencing FL systems. For example, network conditions can play a significant role in multi-server or asynchronous configurations, impacting the overall efficiency and reliability of the system. However, due to the limited timespan of our research, we have chosen to narrow our scope to a single-server synchronous setup and disregard networking limitations for this study. Therefore, we focus on the client and data heterogeneity as the primary factors influencing the FL system performance.

2.2 FL Simulators

FL simulators are software tools and frameworks designed to emulate the FL process. Generally, such simulators can run on a single machine or a cluster of machines, without needing to interact with decentralized devices [7]. To accomplish this, simulators can mock and replicate the behavior of multiple clients and the central server while using synthetic or proxy datasets [6]. Significant effort has been invested into making the upfront cost and effort of developing and deploying real FL systems more manageable, a task in which simulators play an important role. Several novel frameworks have been developed in recent years, of which FedML Parrot [25], PySyft [31], FLUTE [8] and Flower [4] are among the most well established. In the context of this paper, we focus on Flower, a lightweight and flexible open-source FL framework that offers both a deployment and simulation API.

Existing papers on FL simulators have had a strong focus on improving the usefulness of such systems. Three main commonly discussed areas can be identified:

- **Flexibility:** Supporting a wide range of existing FL algorithms, enabling easy benchmarking as done by Baumgart et al. [2]. In addition, easy integration of novel algorithms and configurations by building an extensible framework API has been a key focus of most industry-standard simulators [8; 4; 7; 25; 31].

- **Performance:** While frequently mentioned by papers initially introducing new simulation frameworks, FL simulator performance optimization is also continually being improved by third-party extensions. For example, Protea [29] and Pollen [24] successfully optimize Flower simulations by making the simulations aware of client resources, resulting in faster and more efficient simulations.

- **Ease of Deployment:** In recent years, there has been a greater focus on closing the gap between FL simulation and deployments. An important goal has been to reduce the effort needed for moving an FL configuration from a simulation environment to a real-world deployment. Efforts have been made to enable the migration without laborious code changes, and this has become a key feature of specific FL frameworks such as FedML Parrot [25], NVIDIA FLARE [23] and Flower [4]

However, little effort has been put into understanding if the simulations are accurate representations of real-world deployments. Specifically, there exists a gap in current research in understanding how factors influencing real-world FL deployments, such as data and client heterogeneity, should be reflected in simulations. This involves understanding how the usage of simulators can be improved to better represent real-world conditions, including which system attributes have the most significant impact on the simulator's fidelity and realism. This paper aims to address this gap by identifying key factors present in real-world FL systems that can significantly improve the fidelity of simulations in replicating deployment results.

3 Related Work

While our research topic has not been directly addressed in the literature, several studies have explored related aspects of FL simulations and tracing. In this section, we review the existing work related to our research question and discuss how our study contributes to the current body of knowledge.

Many studies [17; 12; 14; 21; 10] have explored difficulties related to non-IID conditions in FL systems. These studies have shown that the heterogeneous nature of real-world deployments can lead to slower convergence and poor model performance. In the context of these works, two challenges overlapping with our work can be identified: client and data heterogeneity, and the following paragraphs will discuss how our work is unique.

A noteworthy study by Li et al. [17] investigated the impact of non-IID data on FL model performance. This study focused on how volume and label distributions affected the performance of different FL algorithms. While Li et al. examined the impact of these individual non-IID factors in comparison to a fully IID scenario, our research extends this approach in three ways. First, we consider client heterogeneity, which is not accounted for in their study. Second, we compare simulation scenarios with a single non-IID attribute to a pseudo-real deployment with all non-IID factors, providing insight into which factors influence deployments the most. Lastly, Li et al. primarily focus on final performance metrics, while our work critically examines training progressions and convergence.

Regarding client heterogeneity, Gouissem et al. [10] surveyed existing work on incorporating client information into the FL training process. The study highlighted existing techniques for client selection and resource-aware algorithms to optimize performance and convergence speeds. While not directly related to our work, this study provides valuable insights into incorporating client information into FL systems, which is relevant considering our approach to system tracing.

Many tools have been introduced to incorporate information on clients to optimize performance. Of these, FedTrace [30] and Protea [29] are notable examples, each collecting information on factors influencing the clients and introducing algorithms for more effective client selection. While our work similarly examines the effect of non-IID client attributes, we do so in an abstract manner, only exploring higher-level privacy-preserving attributes such as batch size and local epochs, as opposed to their collection of CPU and memory capabilities. Additionally, these tools only aim to improve FL efficiency, not improve the simulation realism compared to deployments.

Lastly, the conceptual idea introduced with FedDebug [9] shares multiple similarities with our work. FedDebug is a debugging tool that *"selectively records an FL application's telemetry data"* [9] to mirror a real FL system. Similar to our work, FedDebug aims to bridge the gap between simulations and real-world deployments by providing a detailed view of the training process, accomplished by collecting both training hyperparameters and individual client models. However, while FedDebug focuses on creating a fully replayable environment and is primarily intended to identify faulty clients, our work focuses on what information is the most important to enhance the accuracy of FL simulations. Therefore, while both explore the use of system traces, the goals and outcomes of the two studies differ significantly.

In summary, most existing work on complementing FL simulations with real-world information has been performance and efficiency-focused. The novelty of our work is therefore in exploring the potential for improving the realism and fidelity when using information from deployments in simulations, and to what extent adding more layers of information benefits that goal.

4 Methodology

This section outlines the methodology of our study, encompassing the systematic approach we take to answer our research question. Section 4.1 describes the reasoning behind the overarching experiment design, while section 4.2 provides a detailed overview of the implementation of the experiment. Following, section 4.3 outlines the independent and dependent variables, and section 4.4 describes the characteristics of the chosen dataset. Lastly, 4.5 outlines the hardware and software components used to run the experiment.

4.1 Experiment Rationale and Goals

To answer our research question, we first make assumptions about the non-IID attributes of clients and data in a deployment. Subsequently, while running the deployment, we capture these distributions and general performance metrics to establish a benchmark for comparison. Establishing a baseline is crucial because we aim to understand how simulators compare to the deployment when additional information about the deployment is gradually included in the simulated environment. In essence, our goal is to investigate which characteristics of the deployment are most useful to harness in simulators to make them more representative of the real world. We believe it is impactful to investigate how simulators can perform more like the deployment, by examining the following simulation configurations:

- **Simulating blindly:** This configuration assumes all client and data attributes are IID, with no specific information from the real deployment. It serves as a baseline to compare against simulations with more detailed deployment data.
- **Simulating with 1 non-IID factor:** Here, the simulation includes one specific non-IID attribute from the deployment, such as non-IID data labels, while other factors remain IID. This isolates the impact of each non-IID factor on the simulation's performance.
- **Simulating with real conditions:** This simulation uses the actual non-IID distributions for all relevant factors observed in the deployment, aiming to replicate the real conditions closely and evaluate the simulator's performance when fully informed about the deployment's heterogeneity.

In summary, these simulations allow us to systematically explore the importance of different non-IID factors in achieving realistic performance outcomes. By incrementally adding

deployment-specific information, we aim to identify the most critical attributes that enhance the accuracy and reliability of FL simulations.

4.2 Experiment Structure and Implementation

Following the established experiment rationale, we now detail how the experiments are implemented and conducted. Overall, the experiment is structured into several components.

First, we run a pseudo-real FL deployment using Flower's deployment API, where the server and clients are executed in separate subprocesses with communication taking place over the localhost network. In the deployment, clients are randomly initialized with heterogeneous properties (detailed in section 4.3) which are saved to disk after training. These properties are then used to replicate the clients in subsequent simulations. The deployment is run for 100 communication rounds, during which centralized (i.e., server-side) metrics are continuously collected to monitor the training process. Additionally, 20 clients are used with a sampling rate of 50% for each round. Communication rounds, number of clients, and sampling rate remain identical for all simulations as well.

Following the deployment, several simulation scenarios are run using Flower's simulation API. Unless explicitly stated, client and data properties are IID for every simulation. Initially, one baseline simulation is run with these purely IID properties, referred to as a 'blind' configuration. Furthermore, as previously outlined, simulations are run with a single non-IID property. Lastly, all non-IID properties are applied to a 'real' simulation.

A full overview of the experiment structure is provided in Table 1. To reduce random noise, all configurations are run 5 times with unique fixed seeds for generating the non-IID client and data distributions.

Experiment	BS	LE	DV	DL
Deployment	Non-IID	Non-IID	Non-IID	Non-IID
Sim. real	Non-IID	Non-IID	Non-IID	Non-IID
Sim. with BS	Non-IID	IID	IID	IID
Sim. with LE	IID	Non-IID	IID	IID
Sim. with DV	IID	IID	Non-IID	IID
Sim. with DL	IID	IID	IID	Non-IID
Sim. blind	IID	IID	IID	IID

Table 1: Comparison of the deployment and simulations.
BS: Batch size, LE: Local epochs, DV: Data volume, DL: Data labels.

An arbitrary convolutional neural network (CNN) is used as the model to train. The CNN architecture consists of two convolutional layers, two max-pooling layers, and three fully connected layers. The full architecture is detailed in Appendix A, Table A1. We use stochastic gradient descent to update the model weights with a learning rate of 0.001 and momentum of 0.9. To aggregate the models, we use the FedAvg [22] algorithm, as this is the most commonly used in recent literature.

4.3 Variables and Evaluation Metrics

The independent variables are categorized into two main groups: training resources and data diversity, intended to reflect the client and data heterogeneity of the system, respectively. Each group consists of two variables that can be adjusted to test different conditions within the FL environment.

Training resources This category focuses on the resources available to each client in the training process. In a deployment, resources can include memory and computing power, each impacting the system in different ways. The following variables are considered as an abstraction of the hardware conditions of the clients and are useful for understanding how non-IID resources affect the training process. The variables include:

- **Batch size**: The number of data samples processed before the model is updated. Varying batch sizes can influence the speed and stability of the training process.
- **Local epochs**: The number of times the model is trained on the entire local dataset before updating the global model. This can affect the convergence rate and accuracy of the model.

In an IID setting, all clients have a default equal batch size of 32 and local epochs of 4. In a non-IID setting, the batch size is randomly chosen from 16, 32, 64, and 128, and the local epochs are randomly chosen from 1, 3, 5, and 7. The non-IID client distributions are visualized in Figure 1.

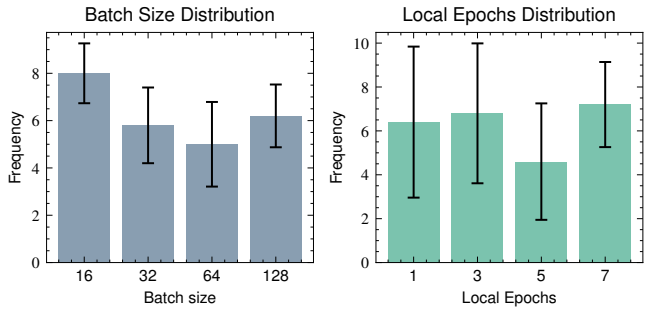

Figure 1: Uniform distribution for non-IID client attribute heterogeneity in a 20-client setting, showing the average and standard deviation over 5-client configurations. Left: Batch size distribution. Right: Local epochs distribution.

Data diversity Data diversity relates to how data is distributed across clients in the FL system, reflecting the heterogeneity of the data each client possesses. Data heterogeneity can include both statistical skewness (i.e., number of data points) and volumetric skewness (i.e., number of labels). The variables include:

- **Data volume**: The number of data points each client has and uses for training. This can affect the model's ability to generalize and learn from different amounts of data.
- **Data labels**: The distribution of data labels across clients. This can impact the model's ability to learn from different classes and generalize to unseen data.

In an IID setting, all clients have an equal amount of data with all labels present. In a non-IID setting, the data volume and labels are sampled from a Dirichlet distribution with an alpha value of 0.5, established to be representative of a real-world distribution [17]. The non-IID data distributions are visualized in Figure 2.

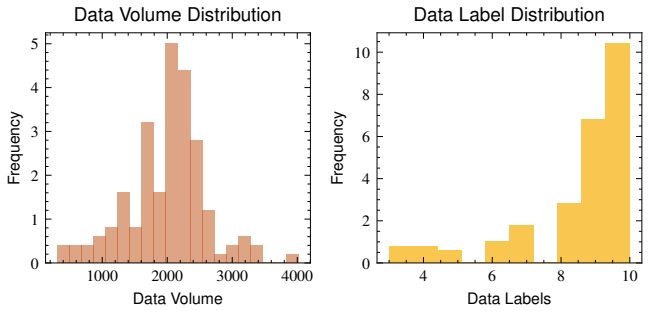

Figure 2: The Dirichlet distribution used for non-IID data heterogeneity in a 20-client setting. Values averaged over 5 configurations. Left: Data volume distribution. Right: Data label distribution.

While generating a Dirichlet distribution, both the data volume and labels are non-IID. However, to isolate the effect of each variable, one must recreate the data volume and labels separately in simulations. In other words, we must be able to have IID volumes with non-IID labels, and vice-versa. This allows for examining how each property influences the simulation independently. To accomplish this, we use the following algorithm (full description can be found in Appendix B, Algorithm 1).

In the `replicate_client_distributions` algorithm, the main objective is to allocate data points and labels to multiple clients, given individual distributions of either an IID or non-IID nature. The process begins by creating a mapping of each label to its corresponding data indices, which are then shuffled to ensure randomness. For each client, a target number of data samples and labels are given. The algorithm ensures that the selected labels are distributed across the client's data points. If the data points are insufficient, additional indices are assigned from a pool of remaining indices. This approach effectively replicates the distribution of data among clients, maintaining the desired IID/non-IID characteristics.

By manipulating these independent variables, we aim to comprehensively assess the accuracy of FL simulations in comparison to actual deployments, providing insights into how different factors influence the performance of FL systems.

Evaluation metrics Several key metrics are employed to evaluate the accuracy of the simulations. These include centralized accuracy, which measures the overall performance of the model on a central test dataset, and cross-entropy loss, which represents the model's error rate. These attributes are gathered throughout the training to assess how quickly the model converges and reaches its optimal performance. Calculating the pair-wise Mean Squared Error (MSE) of the accuracy progression is helpful in quantifying differences between configurations in convergence and performance over time. Combined, these metrics can provide a comprehensive understanding of the fidelity and reliability of FL simulations.

4.4 Dataset

The dataset used in this study is the CIFAR-10 dataset [15], a collection of 60,000 32x32 RGB images in ten classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images. The CIFAR-10 dataset is commonly used in machine learning research and provides a challenging benchmark for image classification tasks. Examples of labels include 'airplane', 'truck', 'bird', and 'cat'.

4.5 Hardware and Software

The deployments were run on a MacBook Pro with an M2 Pro chip (10 cores) and 16GB of memory, running MacOS Sonoma 14.5. All simulations were run on a cloud-based virtual machine instance hosted by Hetzner Cloud[1]. The instance configuration is called *CPX51*, with 16 vCPUs, 32GB of memory, and running Ubuntu 24.04.

For the software stack, the experiments were run using Python 3.9.13, PyTorch 2.2.2, TorchVision 0.17.2, and the Flwr 1.8.0 library for FL deployments and simulations. The code repository can be found on GitHub[2], including the full list of Python packages used.

5 Results

The experiment results reveal several key findings regarding the performance of FL simulations under varying non-IID conditions when compared to the deployment. Following, we highlight the most relevant observations and discuss their significance.

In general, the simulations consistently outperformed the deployment scenario. This aligns with expectations for the simulations with the majority IID conditions, as the non-IID nature of the deployment introduces a level of complexity and variability not present in the more controlled simulation environments. However, a surprising finding was the difference between the deployment and 'real' simulation. Despite both having identical fully non-IID attributes, Table 2 reveals the deployment reached a best accuracy of 53.77% and loss of 1.28, compared to the better 57.12% accuracy and loss of 1.20 for the 'real' simulation. While Figure 3 illustrates that the 'real' simulation had one of the lowest MSE of all configurations relative to the deployment with a value of 21.1, Figure 4 reveals an increasing separation in their performance after the 20th communication round. This contradicts our expectations of simulations and deployments with identical conditions performing similarly. The difference in model convergence highlights a fundamental disconnect between the environment in a simulator and a pseudo-real deployment, and more investigation is needed to further understand this discrepancy.

[1] https://www.hetzner.com/cloud/

[2] https://github.com/Alex-Nygaard/research-project

Configuration	Accuracy	Loss
Deployment	53.77% ± 2.43%	1.28 ± 0.06
Sim. Real	57.12% ± 2.48%	1.20 ± 0.07
Sim. Batch Size	59.09% ± 1.58%	1.15 ± 0.04
Sim. Local Epochs	59.17% ± 2.23%	1.15 ± 0.06
Sim. Data Volume	60.26% ± 0.96%	1.12 ± 0.03
Sim. Data Labels	57.12% ± 1.27%	1.21 ± 0.03
Sim. Blind	59.92% ± 1.42%	1.13 ± 0.05

Table 2: Best model performance metrics for each configuration, averaged over 5 runs.

From Table 2, the 'blind' simulation, with only IID attributes, achieved an accuracy of 59.92% and a loss of 1.13. While among the best-performing configurations, the 'blind' simulation is surprisingly not the best in either metric. Instead, the simulation with non-IID data volumes performed slightly better, with a best accuracy of 60.26% and a loss of 1.12. Several reasons could explain this, such as non-IID data volumes helping prevent overfitting the model. Alternatively, one could argue that the relatively small difference in performance (<1% for accuracy and 0.01 for loss) could be attributed to random noise, with non-IID data volumes playing an insignificant role.

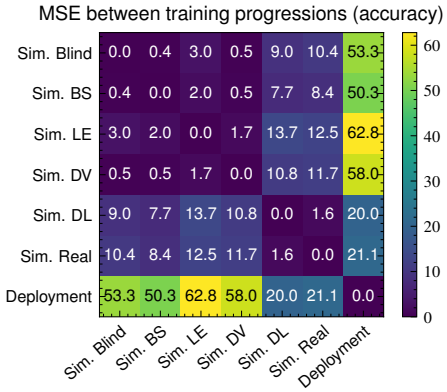

Figure 3: Mean squared error (MSE) between accuracy progressions for each experiment configuration. Accuracies have been scaled from [0,1] to [0,100] during calculation, for readability. Low values mean similar training progression, while higher values indicate larger differences. *BS*: Batch size, *LE*: Local epochs, *DV*: Data volume, *DL*: Data labels.

Another interesting observation is in the similarity between the 'blind' simulation and the simulations with non-IID batch sizes, local epochs, and data volume. As shown in Figure 3, these four simulation configurations progress almost identically, with the highest MSE value of 3.0 between them. Additionally, their best accuracies are all within 1.2% of each other, and losses within 0.03. These metrics could indicate that incorporating non-IID batch sizes, local epochs, and data volume has an insignificant impact on replicating the conditions of the deployment, as the fully IID 'blind' simulation performed very similarly. While Figure 4 reveals that the simulation with non-IID local epochs converged the fastest in the first 40 communication rounds, the performance trends become more uniform after this point.

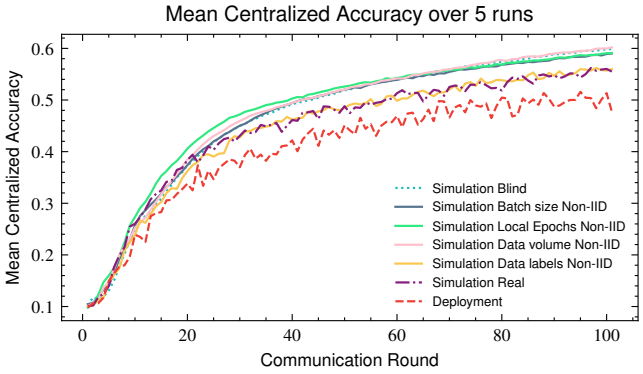

Figure 4: Experiment results showing the centralized accuracy progression over 100 rounds for each of the configurations. Results are averaged over 5 runs, each run with unique seeds for attribute generation.

Furthermore, the simulation with non-IID data labels performed the closest to the deployment in terms of the MSE similarity measure, with an MSE value of 20, as shown in Figure 3. Compared to the 'real' simulation, which achieved an MSE value of 21.1, it becomes clear that these configurations progressed very similarly throughout training, as also shown in Figures 4 and 5. Incorporating non-IID data labels in the simulation also had a significant effect on the best accuracy and loss values, achieving scores of 57.12% and 1.21, respectively. Further comparison to the 'real' simulation in Table 2 shows an identical accuracy and a <1% difference in loss.

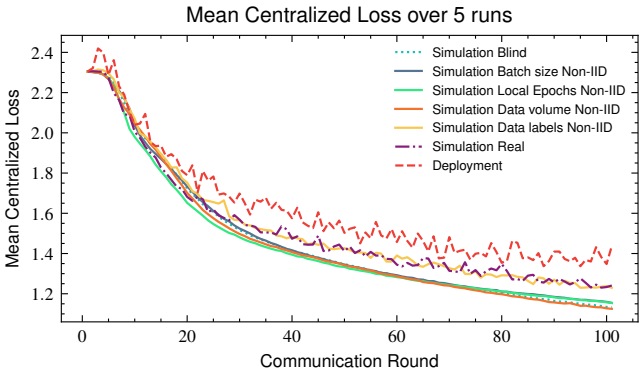

Figure 5: Experiment results showing the centralized loss progression over 100 rounds for each of the configurations. Results are averaged over 5 runs, each run with unique seeds for attribute generation.

Moreover, Figures 4 and 5 illustrate a distinct difference in noise between the training progressions of all configurations including non-IID data labels, compared to the others with IID data labels. This training instability is visible in the deployment, 'real' simulation, and the non-IID data label simulation. In comparison, the other simulations with IID

data labels display smoother convergence curves. This observation implies the noise in the fully non-IID simulation and deployment is a direct result of non-IID data labels, while the other non-IID factors do not have a pronounced impact on the instability of the system.

Summarizing the above insights, we can draw the conclusion that non-IID data labels have the most significant impact on how representative simulations are of deployments, among the examined attributes. From the results of the experiment, we see that the 'blind' simulation performed almost identically to the simulations with non-IID batch sizes, local epochs, and data volumes. Simultaneously, the 'real' simulation performed very similarly to the simulation with non-IID data labels. Combining these two observations, we conclude that incorporating non-IID data labels in simulations is the only factor that improved the realism of the simulation environment, compared to the deployment. Concretely, using non-IID data labels results in a 3.35% difference in accuracy compared to the deployment, a noticeably closer result in comparison to the 'blind' simulation's 6.15% difference. In contrast, the other factors appear to be insignificant in this regard.

In summary, the results of the experiment highlight two key insights. First, there is a noteworthy difference in FL deployments and simulations while using identical client and data configurations. As our experiment setup is insufficient to explain this difference, further investigation is needed in this area. Second, we note the sole impact of incorporating non-IID data labels into simulations. Doing so is the only configuration meaningfully approaching the deployment results, while other factors play a significantly less important role in improving the simulation realism.

6 Discussion

This section critically examines the methodology and results of the study, aiming to identify potential limitations and suggest future work that could address these limitations.

6.1 Limitations

The primary limitation of the methodology lies in the framing of the study's core question, which involves using information from a real FL deployment. Due to the challenges associated with accessing real-world client data, the study utilizes a pseudo-real deployment with certain assumptions. While this approach enables controlled experiments and a clear understanding of the system, it inherently limits the study's applicability to real-world scenarios. The pseudo-realness provides a valuable sandbox environment but lacks the unpredictable nature of actual deployments.

Another methodological constraint is the simplistic FL topology and setup employed in the study. The single-server synchronous model, while effective for controlled experiments, does not reflect the more complex and varied topologies commonly used in the industry or actual FL deployments. Additionally, the limited number of clients sharply contrasts with real-world deployments, which often involve thousands of clients. During the experiment design phase, we planned to include more clients to better represent real-world scenarios. However, constraints related to available computing resources and financial limitations prevented us from scaling up. This discrepancy underscores the need for more scalable experimental setups to better mimic real-world conditions.

The results of the study also reveal several limitations. The scope of independent variables is relatively narrow, which could affect the generalizability of the simulation outcomes. There are numerous other factors that could potentially influence the performance of FL systems, and excluding these from the study represents a significant gap. Furthermore, the dependent variables measured are not exhaustive. While the study focuses on centralized accuracy and loss, other critical metrics such as decentralized performance metrics, communication overhead, and convergence time, are not considered. This limited view may overlook important aspects of FL system performance.

6.2 Future Work

To address these limitations, future work should explore extending the configurations studied to include asynchronous models and investigate the impact of network factors. Asynchronous models are more reflective of real-world FL systems and can provide insights into performance under varied network conditions and client availability.

Additionally, developing more advanced techniques to collect trace information in a privacy-preserving manner is crucial. Such techniques would enable the gathering of more accurate and granular data from real-world deployments without compromising client privacy. This advancement could significantly enhance the realism of simulators and the applicability of FL research.

Future research should also aim to trace FL performance in a real system or orchestrate more realistic pseudo-real experiments involving actual edge devices, such as phones, IoT devices, and servers. Additionally, exploring the creation and use of proxy datasets to mirror real data trends would provide significant value. By combining more representative client configurations and proxy data, researchers can better validate their findings and ensure that their models accurately reflect real-world conditions. This approach would bridge the gap between controlled experimental setups and the unpredictable nature of real-world FL deployments, providing more robust and generalizable insights into FL system performance.

Lastly, as previously mentioned, more investigation is also needed to understand the discrepancy between simulations and deployments with identical configurations.

7 Responsible Research

It is crucial to analyze and understand any potential ethical implications of the research conducted. This section will discuss the potential risks and limitations of the research, as well as the steps taken to mitigate these risks.

While FL is a highly privacy-oriented field of study, the research conducted as a part of this paper does not involve any sensitive real-world data. Instead, all data used is publicly available and synthetic, and commonly used in the existing literature. Therefore, no personal data is used in this research, and the privacy of individuals is not at risk.

Reproducibility is a key aspect of research, and we have attempted to make our work as reproducible as possible. All code used in the experiments is available on GitHub[3], and the data is publicly available. While seeds are used in pseudo-random number generation (e.g., to generate client configurations), due to inherent non-determinism in multi-threaded/process code execution, it is possible that results may vary slightly between runs. For example, the order in which clients are sampled and updates are processed may slightly affect the final model. However, the overall trends and conclusions should remain consistent, especially when averaging results.

Moreover, all non-cited concepts shared in this paper are independently generated by the authors without external collaboration. While appendix section C outlines the use of LLMs as brainstorming partners and writing assistants, all generated ideas in this paper are products of the authors' work alone.

In conclusion, the research involves low risk, is highly reproducible, and is independently conducted, satisfying TU Delft's ethical guidelines for research.

8 Conclusion

In this study, we have explored the potential of using real-world FL deployment traces to improve the accuracy of simulation environments. This was accomplished by examining the impact of non-IID conditions within an FL system, specifically the attributes batch size, number of local epochs, data volume, and data label distributions. By comparing the fully non-IID pseudo-real deployment to varying simulations, each incorporating different IID and non-IID attributes, we can point to several key findings.

First, the results of the experiment indicate there is a fundamental difference in the performance of a non-IID deployment and an identical simulated system. The reasons for this can be complex, and further investigation has been identified as a future direction to extend this study. Another key insight is the significantly stronger influence that data label distributions have on how accurately simulations recreate FL deployment results when compared to the other examined attributes. The experiment outcomes showed that simulations incorporating non-IID labels for clients most closely match the learning progression of deployments. Additionally, incorporating non-IID labels can be identified as the largest source of noise in the learning progression, as all simulations with non-IID labels undergo significantly more unstable training when compared to IID label configurations. Moreover, the results of the experiment also show that the outcome of the simulations does not change significantly when introducing non-IID batch sizes, number of local epochs, and data volumes.

It is important to mention the limitations of the aforementioned findings. The experiment structure and scope inherently introduce limitations, amongst which the scale and pseudo-realness can be identified as the most impactful. First, the experiments were run at a smaller scale compared to traditional FL system seen in commercial or academic environments. Therefore, our findings can only be regarded as significant for small-scale FL systems, and future work is needed to explore the impact of using more clients. Second, the pseudo-real nature of the deployment is a limiting factor, as the system configuration is based on assumptions about commercial deployments, as opposed to actual live systems. Therefore, our experiments have limited practical applications, as using real-life system traces would provide better conclusions.

Regardless, the presented results are significant because they can provide valuable insight into what improvements can be made in the process of developing FL systems at scale. We demonstrate the value that collecting and incorporating specific information from deployments into simulations can provide. These insights can have practical applications in both academic and commercial settings. For academia, more realistic benchmarks can be created to represent the complexity of the real world. For commercial applications, maximizing the usefulness of client data collected from live FL systems while minimizing the impact on user privacy is important, and can be accomplished by only tracing the most influential attributes. To this extent, building a better understanding of how more realistic development environments can be created is crucial, as it will improve the process of creating helpful and safe decentralized machine learning systems.

[3]https://github.com/Alex-Nygaard/research-project

Appendix A Neural Network Architecture

Layer Type	Input Size	Output Size
Conv2d	3 x 32 x 32	6 x 28 x 28
ReLU	6 x 28 x 28	6 x 28 x 28
MaxPool2d	6 x 28 x 28	6 x 14 x 14
Conv2d	6 x 14 x 14	16 x 10 x 10
ReLU	16 x 10 x 10	16 x 10 x 10
MaxPool2d	16 x 10 x 10	16 x 5 x 5
Flatten	16 x 5 x 5	400
Linear	400	120
ReLU	120	120
Linear	120	84
ReLU	84	84
Linear	84	10

Table A1: Architecture of the CNN to be used in the experiments.

Appendix B Algorithms

Algorithm 1 *replicate_client_distributions* - Given data volume and label counts for clients, replicate the distributions with different dataset indices.

Require: $datapoints\_per\_client$, $labels\_per\_client$
Ensure: $idx\_clients$

$prng \leftarrow$ np.random.default_RNG($SEED$)
Create label-to-indices mapping $label\_indices$
for each $indices$ in $label\_indices$ **do**
  Shuffle $indices$ using $prng$
end for
Initialize empty client indices $idx\_clients$
for each $client\_id, (num\_samples, num\_labels)$ **do**
  Select num_labels random labels
  $client\_indices \leftarrow$ empty list
  for each $label$ **do**
    Calculate required samples $count\_needed$
    $count\_assigned \leftarrow 0$
    while indices available and $count\_assigned < count\_needed$ **do**
      Assign indices to client
      Increment $count\_assigned$
    end while
  end for
  Shuffle $client\_indices$ and assign to $idx\_clients[client\_id]$
end for
Collect remaining indices and shuffle
for each $client\_id$ **do**
  if client needs more indices **then**
    Assign from remaining indices
  end if
end for
return $idx\_clients$

Appendix C Use of LLMs

In conducting this research, we utilized OpenAI's ChatGPT, a large language model (LLM), to facilitate various aspects of the research process. The primary functions of ChatGPT in this research included idea discussion and writing assistance. Specifically, ChatGPT was used in the following ways:

Idea Discussion: To enhance the depth and breadth of the research, I engaged ChatGPT in brainstorming sessions to generate and refine ideas. This involved posing questions about potential research directions, exploring various hypotheses, and discussing theoretical frameworks. For instance, questions such as "What are the potential impacts of X on Y?" and "How can theory Z be applied to this research context?" were used to stimulate and broaden the scope of the research inquiry.

Grammatical Writing Assistance: ChatGPT was also employed to improve the clarity and coherence of the written content. This included proofreading drafts, suggesting alternative phrasings for complex ideas, and ensuring grammatical accuracy throughout the manuscript. Queries like "Can you rephrase this paragraph for better clarity?" and "Is this sentence grammatically correct?" were typical examples of how ChatGPT was used to enhance the quality of writing.

It is important to note that while ChatGPT provided significant support in these areas, all final decisions regarding the content and direction of the research were made independently by the author. The use of ChatGPT was supplementary, aimed at improving the efficiency and quality of the research process without compromising the originality and integrity of the work.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

[2] Gustav A. Baumgart, Jaemin Shin, Ali Payani, Myungjin Lee, and Ramana Rao Kompella. Not all federated learning algorithms are created equal: A performance evaluation study, 2024.

[3] Jannis Beese, M. Kazem Haki, Stephan Aier, and Robert Winter. Simulation-based research in information systems. *Business & Information Systems Engineering*, 61(4):503–521, 2019.

[4] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[5] Subrato Bharati, M. Rubaiyat Hossain Mondal, Prajoy Podder, and V. B. Surya Prasath. Federated learning: Applications, challenges and future directions. *International Journal of Hybrid Intelligent Systems*, 18(1-2):19–35, 2022.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 374–388, 2019.

[7] Konstantin Burlachenko, Samuel Horváth, and Peter Richtárik. FL_PyTorch. In *Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning*, New York, NY, USA, December 2021. ACM.

[8] Mirian Hipolito Garcia, Andre Manoel, Daniel Madrigal Diaz, Fatemehsadat Mireshghallah, Robert Sim, and Dimitrios Dimitriadis. Flute: A scalable, extensible framework for high-performance federated learning simulations. *arXiv preprint arXiv:2203.13789*, 2022.

[9] Waris Gill, Ali Anwar, and Muhammad Gulzar. Feddebug: Systematic debugging for federated learning applications, 01 2023.

[10] Ala Gouissem, Zina Chkirbene, and Ridha Hamila. A comprehensive survey on client selections in federated learning. *arXiv preprint arXiv:2311.06801*, 2023.

[11] Badra Souhila Guendouzi, Samir Ouchani, Hiba EL Assaad, and Madeleine EL Zaher. A systematic review of federated learning: Challenges, aggregation methods, and development tools. *Journal of Network and Computer Applications*, 220:103714, 2023.

[12] Venkataraman Natarajan Iyer. A review on different techniques used to combat the non-iid and heterogeneous nature of data in fl. *arXiv preprint arXiv:2401.00809*, 2024.

[13] Yae Jee Cho, Samarth Gupta, Gauri Joshi, and Osman Yağan. Bandit-based communication-efficient client selection strategies for federated learning. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 1066–1069, 2020.

[14] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[16] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers and Industrial Engineering*, 149:106854, 2020.

[17] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978, 2022.

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

[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

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

[21] Ibrahim Abdul Majeed, Hwang-Ki Min, Venkata Siva Kumar Tadi, Sagar Kaushik, Aniruddha Bardhan, Karthikeyan Kumaraguru, and Rajasekhara Reddy Duvvuru Muni. Factors influencing cost and performance of federated and centralized machine learning. In *2022 IEEE 19th India Council International Conference (INDICON)*, pages 1–6, 2022.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[23] Holger R. Roth, Yan Cheng, Yuhong Wen, Isaac Yang, Ziyue Xu, Yuan-Ting Hsieh, Kristopher Kersten, Ahmed Harouni, Can Zhao, Kevin Lu, Zhihong Zhang, Wenqi Li, Andriy Myronenko, Dong Yang, Sean Yang, Nicola Rieke, Abood Quraini, Chester Chen, Daguang Xu, Nic Ma, Prerna Dogra, Mona Flores, and Andrew Feng. Nvidia flare: Federated learning from simulation to real-world. 2022.

[24] Lorenzo Sani, Pedro Porto Buarque de Gusmão, Alex Iacob, Wanru Zhao, Xinchi Qiu, Yan Gao, Javier Fernandez-Marques, and Nicholas Donald Lane. Pollen: High-throughput simulation of federated learning via resource-aware client placement, 2024.

[25] Zhenheng Tang, Xiaowen Chu, Ryan Yide Ran, Sunwoo Lee, Shaohuai Shi, Yonggang Zhang, Yuxin Wang, Alex Qiaozhong Liang, Salman Avestimehr, and Chaoyang He. Fedml parrot: A scalable federated learning system via heterogeneity-aware scheduling on sequential and hierarchical training, 2023.

[26] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535, 2023.

[27] Jiajun Wu, Fan Dong, Henry Leung, Zhuangdi Zhu, Jiayu Zhou, and Steve Drew. Topology-aware federated learning in edge computing: A comprehensive survey. *ACM Comput. Surv.*, apr 2024. Just Accepted.

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

[29] Wanru Zhao, Xinchi Qiu, Javier Fernandez-Marques, Pedro P. B. de Gusmão, and Nicholas D. Lane. Protea:

client profiling within federated systems using flower. In *Proceedings of the 1st ACM Workshop on Data Privacy and Federated Learning Technologies for Mobile Edge Network*, FedEdge '22, page 1–6, New York, NY, USA, 2022. Association for Computing Machinery.

[30] Zirui Zhu and Lifeng Sun. Federated trace: A node selection method for more efficient federated learning. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1234–1238, 2021.

[31] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, Théo Ryffel, Zarreen Naowal Reza, and Georgios Kaissis. *PySyft: A Library for Easy Federated Learning*, pages 111–139. 06 2021.