IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 August 2024, accepted 24 August 2024, date of publication 28 August 2024, date of current version 6 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3450894

RESEARCH ARTICLE

Demystifying Impact of Key Hyper-Parameters in Federated Learning: A Case Study on CIFAR-10 and FashionMNIST

MAJID KUNDROO, (Graduate Student Member, IEEE), AND TAEHONG KIM, (Senior Member, IEEE)
School of Information and Communication Engineering, Chungbuk National University, Cheongju 28644, Republic of Korea

Corresponding author: Taehong Kim (taehongkim@cbnu.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), Ministry of Education under Grant NRF-2022R1I1A3072355, 50%; and in part by the Innovative Human Resource Development for Local Intellectualization Program through the Institute of Information and Communications Technology Planning and Evaluation (IITP), Korean Government (MSIT) under Grant IITP-2024-2020-0-01462, 50%.

ABSTRACT Federated Learning (FL) has emerged as a promising paradigm for privacy-preserving distributed Machine Learning (ML), enabling model training across distributed devices without compromising data privacy. However, the impact of hyper-parameters on FL model performance remains understudied and most of the existing FL studies rely on default or out-of-the-box hyper-parameters, often leading to suboptimal convergence. This study specifically investigates the intricate relationship between key hyper-parameters—learning rate, epochs per round, batch size, and client participation ratio (CPR)—and the performance of FL models on two distinct datasets: CIFAR-10 using ResNet-18 and FashionMNIST using a simple CNN model. Through systematic exploration on these datasets, employing a centralized server and 200 clients, we elucidate the significant impact of varying hyper-parameters. Our findings underscore the importance of dataset-specific hyper-parameter optimization, revealing contrasting optimal configurations for the complex CIFAR-10 dataset and the simpler FashionMNIST dataset. Additionally, the correlation analysis offers a deep understanding of hyper-parameter inter-dependencies, essential for effective optimization. This study provides valuable insights for practitioners to customize hyper-parameter configurations, ensuring optimal performance for FL models trained on different types of datasets and provides a foundation for future exploration in hyper-parameter optimization within the FL domain.

INDEX TERMS Communication cost, federated learning, hyper-parameter optimization.

I. INTRODUCTION

Traditional ML approaches collect data from multiple sources and aggregate it on a central server for model training. However, this centralized data collection raises significant privacy concerns, especially when dealing with sensitive or confidential data [1]. To effectively handle these privacy concerns, FL was proposed by Google in 2016 [2] and has rapidly risen as a promising paradigm for training ML models in a decentralized manner while preserving data privacy [3]. FL addresses these concerns by enabling collaborative model training without the need for directly sharing raw data [1]. Instead of centralizing data, the FL framework involves training models locally on devices or clients where the data is generated. This decentralized approach eliminates the need for data transmission or centralized storage, mitigating potential privacy risks associated with data breaches or unauthorized access [4], [5]. As a result, FL has gathered significant attention in applications where data privacy is a critical concern, such as healthcare, finance, and personalized user experiences on mobile devices [6].

Despite the numerous advantages of FL due to its decentralized nature, it introduces additional complexities compared to centralized ML as well. Factors such as

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

system heterogeneity across clients, statistical heterogeneity, communication constraints, and the inherent dynamics of distributed learning pose unique challenges, especially while selecting hyper-parameters [7]. The decentralized nature of FL introduces system heterogeneity, where clients may have varying computational capabilities, memory constraints, and diverse hardware specifications. Additionally, statistical heterogeneity arises due to the non-independent and identically distributed (non-IID) nature of data across clients, resulting in different data distributions [8]. These challenges necessitate careful consideration while selecting hyper-parameters, as sub-optimal choices can lead to convergence issues, degraded performance, and increased communication overhead [9].

In many FL studies, a common yet ineffective strategy is the adoption of default out-of-the-box hyper-parameters, which are typically optimized for centralized settings [10]. While this approach simplifies the training process, it often proves to be ineffective within the FL context [11]. Ill-suited hyper-parameters can lead to a number of issues, including slow convergence rates, unstable training dynamics, sub-optimal model accuracy, and inefficient resource utilization [12]. Conversely, meticulously tuned hyper-parameters can unlock the full potential of FL models, enabling faster convergence, improved generalization capabilities, and enhanced overall performance [13]. This, in turn, translates into tangible benefits, such as reduced computational costs, improved model deployment times, and better-quality predictions, which are critical for real-world applications [11].

Furthermore, the inter-dependencies between various hyper-parameters in FL settings necessitate a comprehensive understanding of their intricate relationships and trade-offs. Failure to account for these inter-dependencies can result in configurations that appear optimal in isolation but ultimately under-perform when combined [14]. Therefore, a holistic approach to hyper-parameter optimization, involving systematic exploration and analysis of these inter-dependencies, is essential for realizing the full potential of FL models.

This study aims to conduct an analysis of hyper-parameter relationships and their impact on the performance of FL models using CIFAR10 and FashionMNIST dataset. Specifically, we focus on key hyper-parameters including learning rate, epochs per round, batch size, and CPR. The overarching objective is to elucidate the intricate relationships and inter-dependencies among these hyper-parameters and their influence on the convergence dynamics and overall performance of models across datasets of different complexities. By delving into the complexities of hyper-parameter optimization in the FL domain, we aim to contribute significantly to improving model performance by ensuring that FL models are trained using optimal hyper-parameters. Through a series of systematic experiments, we seek to enhance our understanding of the impact of hyper-parameter choices and their inter-dependencies on model performance and datasets, providing valuable guidance for practitioners in optimizing FL systems.

This study holds significant importance and timeliness in advancing the field of FL by addressing a crucial aspect of model optimization – hyper-parameter tuning. By shedding light on the complex relationships between key hyper-parameters, this research paves the way for improved model convergence strategies, enabling more efficient and effective training and deployment of FL in diverse applications.

The key contributions of this study are as follows:

1) **Detailed analysis of hyper-parameter relationships in FL across diverse datasets:** Through systematic experimentation and rigorous evaluation on both CIFAR-10 and FashionMNIST datasets, this study provides a in-depth analysis of the intricate relationships between learning rate, epochs per round, batch size, and CPR. This analysis unveils their collective impact on FL model performance across datasets of varying complexity.
2) **Insights into enhancing model convergence and mitigating associated challenges:** By unraveling the inter-dependencies among these critical hyper-parameters for both complex (CIFAR-10) and simpler (FashionMNIST) datasets, the study offers valuable insights into mitigating convergence challenges and optimizing the training process in FL settings for a range of data complexities.
3) **Comparative analysis of hyper-parameter dynamics across datasets:** The study highlights how optimal hyper-parameter configurations can vary significantly between datasets of different complexities, emphasizing the importance of dataset-specific tuning in FL systems.
4) **Correlation analysis to understand hyper-parameter inter-dependencies:** Through a detailed correlation analysis, the study uncovers the inter-dependencies between various hyper-parameters, providing a deeper understanding of their intricate relationships and trade-offs.

The rest of this paper is organized as follows: Section II introduces related work. Section III discusses an overview of FL, hyper-parameters, and their significance in FL theoretically. The methodology is presented in Section IV along with the used experimental setup and overview of the methodology adapted. In Section V, we present analysis and discussion of the results. Finally, the paper is concluded with conclusions and future scope in Section VI.

II. RELATED WORK

In 2016, FL was proposed as a privacy-preserving approach to train ML models on decentralized data by Konečný et al. [2]. This groundbreaking work laid the foundation for a new paradigm in distributed learning, where data remains localized on individual devices, addressing critical privacy concerns. Despite its benefits, FL presents several challenges, including communication overhead, data heterogeneity, and maintaining model performance across diverse datasets [15] Since its inception, numerous federated

optimization algorithms have been developed to enhance the efficiency and performance of FL. One of the earliest and most fundamental algorithms is FedAvg [6], which has been designed specifically for communication-efficient training in FL settings. This algorithm introduced a novel approach to aggregate model updates from distributed clients, facilitating collaborative learning while minimizing communication overhead.

Building upon the success of FedAvg, researchers have explored various techniques to further improve the performance of FL systems on both server side [16], [17] and client side [18], [19]. Many approaches have been employed on the server side to enhance convergence and address heterogeneity in the data. Reddi et al. [16] proposed the use of adaptive optimization methods, such as ADAGRAD [20], YOGI [21], and ADAM [22], to improve the convergence of federated models, particularly in scenarios with heterogeneous data distributions across clients. The effect of different optimizers, Adam [22] and SGD [23], are analyzed in [24], where they focused solely on the effects of Adam and SGD in FL. Xin et al. proposed FedSSO where they used a server-side second-order optimization method to improve the performance of FL. In addition to server-side optimizations, several approaches have also been employed on the client side like [11], [18], [25], and [26] and many more. One notable example that improves on both client and server side is MOCHA [27], a communication-efficient optimization method that trains distinct yet related models for each device using a multi-task learning framework. This approach aims to overcome communication-related challenges inherent in FL by leveraging the shared representations across tasks, leading to more efficient model updates and reduced communication costs.

As the field of FL continues to evolve, researchers have investigated the impact of various hyper-parameters like learning rate, number of epochs, batch size etc on the performance and convergence of federated models. Learning rate was shown to have a significant effect on the performance of ML models [28] and is expected to impact the performance of FL models as well. In this regard, Koskela and Honkela [18] conducted an analysis of the effects of different learning rates in the context of FL, providing valuable insights into this critical hyper-parameter. Brendan McMahan et al. [6] also compared the fixed and adaptive learning rates in FL settings highlighting the importance of dynamically adjusting learning rate to account for the distributed and non-iid nature of data in FL systems. Moreover, they also explored a variety of learning rates for FedAvg and FedSGD to check the influence of different learning rates on the performance. Reddi et al. found that lower learning rates may help achieve more stable and accurate models, especially in heterogeneous environments [16].

Number of epochs is another significant hyper-parameter that impacts the performance of ML algorithms in general [29], [30] and its effect has also been seen in FL as well. The results in [19] show that increasing the number of local epochs from 1 to 16 per round can significantly improve the convergence speed of the federated model over communication rounds. However, [31] demonstrates that very high epoch values (e.g. 20 epochs) combined with large batch sizes can lead to poor final performance, highlighting the need to carefully tune these hyper-parameters, which also implies the need to study these hyper-parameters together.

Similarly, batch size not only helps in managing the memory but is also known to impact the convergence in ML [32], [33] as well as in FL models. Liu et al. [25] quantified and leveraged the interplay of the number of local update steps and heterogeneous batch sizes across clients for FL and showed larger batch sizes can reduce the number of communication rounds needed, thereby improving communication efficiency. Study like [34] shows that smaller batch sizes generally promote stable convergence and better generalization in FL, similar to centralized training. Conversely, very large batch sizes can cause issues like divergence, poor generalization, and slower convergence times in FL settings. One study suggests batch sizes in the range of 32-64 can strike a good balance between convergence speed and stability for common FL tasks [35]. In addition to the common hyper-parameters between FL and ML, many hyper-parameters specific to FL, could also potentially effect the convergence. However, the effects of such hyper-parameters on convergence has not been studied extensively yet and need further investigations. Our survey revealed only few studies [26], [36] explored the influence of CPR and learning rates to shed the light on the interplay between these factors and their impact on model performance.

Besides, the individual effects of hyper-parameters on model convergence, only few studies have attempted to find the interplay between two or more hyper-parameters. These include works by Khodak et al. [11] who incorporated different learning rates and epoch sizes in their study, although they did not dig into a comprehensive analysis or discussion of their effects. FedTune [36], on the other hand, explicitly investigated the impact of varying the number of clients and epochs, contributing to a deeper understanding of these hyper-parameters in FL settings. Additionally, Kuo et al. [37] explored the role of learning rates and batch sizes, further enriching the body of knowledge surrounding hyper-parameter optimization in FL. Different epochs and batch sizes are also evaluated by [38] in shallow details. Table 1 presents a summary of key hyper-parameters used across various studies, including learning rate, epochs per round, batch size, and CPR.

Based on the survey of literature, we found that none of the studies has specifically targeted to study the cause and effect relation of all the important hyper-parameters in FL. While certain studies cursorily touched the subject of learning rates or epochs, a in-depth analysis of other critical hyper-parameters remains prominently absent. This

TABLE 1. Summary of the hyper-parameters used in literature.

Ref.	Year	Learning Rate	Epochs per Round	Batch Size	CPR
[6]	2017	✓	✗	✗	✗
[18]	2018	✓	✗	✗	✗
[19]	2019	✗	✓	✗	✗
[26]	2019	✓	✗	✗	✓
[16]	2020	✓	✗	✗	✗
[11]	2021	✓	✓	✗	✗
[31]	2021	✗	✓	✗	✗
[34]	2021	✗	✗	✓	✗
[35]	2021	✗	✗	✓	✗
[36]	2022	✗	✓	✗	✓
[25]	2023	✗	✗	✓	✗
[37]	2023	✓	✗	✓	✗
[38]	2023	✗	✓	✓	✗
This study		✓	✓	✓	✓

gap in the existing literature became the the motivation for our endeavor to undertake an exhaustive exploration of prominent hyper-parameters in FL, analyzing their complex relationships with model performance.

III. PRELIMINARIES

A. FEDERATED LEARNING

FL represents a decentralized approach for training ML models by utilizing extensive distributed datasets. This method ensures the privacy of data during training by keeping datasets on clients, owned by organizations or individuals who prefer not to disclose their information. Under centralized supervision, models are trained locally on these clients. Periodically, a server collects the learned parameters to update the global model, subsequently distributing it back to clients for local training and inference. The training process in FL typically occurs in rounds, each comprising of the following steps.

1) FL server chooses a global ML model that is to be trained on the client's data.
2) A subset of clients is randomly selected and the global model is broadcasted to the selected clients.
3) The clients upon receiving the global model, trains the model using its local data to update its weights and then transmits the updated weights to the server.
4) The server receives the updated model weights from all clients and aggregates them to construct a new global model.
5) The new global model is then again broadcasted to all the clients and these steps continue till the model converges.

FL aims to achieve model improvement while preserving privacy by keeping data localized this decentralized training methodology is particularly useful in scenarios where data privacy is a priority or when centralizing large datasets is impractical.

B. HYPER-PARAMETERS AND THEIR SIGNIFICANCE IN FL

In the realm of ML, hyper-parameters refer to external configurations or settings predetermined before the commencement of the training process. These parameters, distinct from those learned from data, are defined by practitioners or determined through optimization techniques. Hyper-parameters play a crucial role in shaping the behavior and performance of the learning algorithm [39]. A hyper-parameter, identified as a variable that significantly influences the learning process by affecting the parameters or weights learned by a ML algorithm, is aptly termed as such due to its distinctive role at the "top level" of the learning hierarchy. This nomenclature emphasizes its pivotal position with a profound impact on the overarching learning process and the subsequent determination of model parameters. The meticulous selection of these hyper-parameters is important, as it directly shapes the trajectory and speed of the learning algorithm. Recognizing the importance of hyper-parameters is integral to achieving optimal model performance and ensuring effective training outcomes in ML applications. Several critical hyper-parameters, involved in FL and their significance is given below:

1) LEARNING RATE

The learning rate, a critical hyper-parameter, that plays a pivotal role in determining the size of steps taken during the optimization process and influences how quickly or slowly a model converges to the optimal set of parameters during training [28]. Represented as a scalar value, the learning rate scales the gradient descent or optimization updates applied to the model weights. The significance of the learning rate includes:

1) Affects model convergence: An appropriately set learning rate enables efficient navigation of the error surface to reach an optimal or near-optimal minimum. Too small a rate slows this journey down while too large a rate overshoots the destination [29].
2) Impacts training stability: Choosing the right learning rate is crucial to ensure training stability and avoid fluctuations or divergent behavior. Overlarge rates tend to cause large weight updates leading to oscillating losses. Appropriate rates maintain steady smooth convergence.

Therefore, choosing an optimal learning rate is crucial for both model convergence and performance. The learning rate is a crucial factor that affects the convergence, stability, and efficiency of training ML models. Striking the right balance by carefully tuning the learning rate is essential for achieving optimal performance and ensuring that the model generalizes well to unseen data.

2) NUMBER OF EPOCHS

The term "epoch" in ML refers to one complete iteration of training a model. The number of complete runs required to prepare an algorithm efficiently is specified in terms of epochs, and the model's internal parameters are modified after each epoch. Epochs per round is an important hyper-parameter in FL that controls how many local epochs or training iterations each client performs on their local data

before sending an update to the central server. The number of epochs needed is related to the size of the dataset and learning rate [30]. As the size of the dataset increases, the number of epochs required for convergence also increases with respect to the specified learning rate [29]. The significance of epochs per round in FL includes:

1) Controls client computation vs. communication trade-off: More local epochs means clients do more computation in parallel before communication. This reduces communication overhead but can lead to client models diverging too much.
2) Affects model performance: Generally, more local epochs leads to better client model performance up to a point before overfitting. But too many epochs may cause client model divergence.
3) Impacts system efficiency: With more compute intensive clients, fewer local epochs may allow involving more available clients in each round at the expense of potentially decreased model accuracy.

In summary, determining the optimal number of epochs per round in FL involves considering trade-offs between training efficiency, model performance, communication overhead, and global model consistency. Careful tuning of this hyper-parameter is crucial for achieving efficient and effective FL outcomes.

3) BATCH SIZE

Batch size is also one of the most important hyper-parameters and it represents the number of samples used in one forward and backward pass through the network. Batch size controls the amount of local data used in each training iteration on the clients. The significance of batch size includes:

1) Affects model accuracy: Larger batch sizes tend to improve model accuracy, but can reach a plateau or even diminish returns after a point.
2) Impacts computational efficiency: Larger batches allow more parallelism and GPU utilization, leading to faster training time. However, extremely large batches may not improve throughput and can even degrade performance.
3) Controls client memory usage: Larger batches require more temporary memory during client gradient computation. More resource-constrained clients may only support smaller batches.

Overall, the batch size in FL is a crucial hyper-parameter that requires careful consideration. It involves balancing communication efficiency, global model consistency, local model generalization, and resource constraints. The optimal batch size is often determined through experimentation and fine-tuning to achieve efficient and effective FL outcomes.

4) CLIENT PARTICIPATION RATIO

CPR in FL refers to the fraction or percentage of total clients that actively participate in a specific training round. It is a hyper-parameter that determines the proportion of clients whose local models contribute to the global model update during a FL round. Its significance includes:

1) Affects model performance: Higher participation allows aggregating updates from more clients, improving accuracy and generalization. But too high a ratio can diminish returns.
2) Impacts training efficiency: Lower participation performs more local epochs on each selected client. This enables faster rounds, but can reduce accuracy if too few clients participate.
3) Relates to system scalability: Participation ratio must balance with increase in number of clients. As more clients enroll, keeping participation ratio constant allows scaling up while maintaining statistical significance.
4) Controls failure tolerance: When participation is low, failure of few clients has higher impact. Higher ratios make the system more fault tolerant.

An ideal participation ratio ensures statistical significance while maximizing accuracy and system efficiency. Values typically range from 5-50% depending on the number of clients and their capabilities. CPR in FL is a critical parameter that impacts communication efficiency, resource utilization, convergence dynamics, model robustness, and generalization capabilities. Careful tuning of this hyper-parameter is essential to achieve a balance that aligns with the goals and constraints of the FL scenario at hand.

IV. METHODOLOGY

In this study, our main objective is to elucidate the influence of crucial hyper-parameters within the complex landscape of FL systems. We focus specifically on four pivotal hyper-parameters - the learning rate, epochs per round, batch size, and CPR. There are other hyper-parameters as well involved in FL process, but in this study we are focusing only on four important hyper-parameters which are mentioned above. Comprehending the roles played by these hyper-parameters and their effects is fundamental to the overarching goal of enhancing model performance and expediting training convergence.

To conduct our investigation, we designed an experimental FL framework including datasets and model architectures discussed in subsection IV-A and subsection IV-B provides an overview of our experiments.

A. EXPERIMENTAL SETUP

1) FL FRAMEWORK

For this study, a federated framework is developed using Flower (v1.15) [40], that allows for flexible and scalable FL simulations. Flower was chosen for its ease of use and ability to simulate a large number of clients. The framework was set up with a central server and 200 client devices, enabling the simulation of a realistic FL environment.

2) DATASETS

- **CIFAR-10 [41]:** A widely used dataset in computer vision, consisting of 60,000 32×32 color images across 10 classes. The dataset was split into 50,000 training images and 10,000 test images. For our federated learning setup, the training data was evenly distributed among the 200 clients in an IID manner.
- **FashionMNIST [42]:** A dataset of Zalando's article images, consisting of 70,000 28×28 grayscale images across 10 fashion categories. It includes 60,000 training images and 10,000 test images. Similar to CIFAR-10, the training data was evenly distributed among the 200 clients in an IID manner for the FL experiments.

The rationale behind using two datasets, CIFAR-10 and FashionMNIST, for this experiment is to analyze the effects of hyper-parameters in FL across different levels of dataset complexity. CIFAR-10, a more complex dataset with color images of various objects, contrasts with FashionMNIST, which consists of simpler grayscale images of clothing items. This diversity allows for a robust evaluation of how hyper-parameter settings need to be adjusted based on dataset characteristics, ensuring the findings are broadly applicable and not limited to a single type of data.

3) MODEL ARCHITECTURES

- **Resnet-18:** For the more complex CIFAR-10 dataset, we employed the ResNet18 [43] architecture. ResNet18 is a deep convolutional neural network known for its ability to handle complex image classification tasks. It consists of 18 layers and utilizes skip connections to mitigate the vanishing gradient problem, making it well-suited for the diverse and challenging nature of the CIFAR-10 dataset.
- **Simple CNN:** For the simpler FashionMNIST dataset, we used a lightweight convolutional neural network taken from [6]. This CNN architecture consists of two convolutional layers followed by max pooling, a flattening operation, and two dense layers. It is designed to be efficient for the less complex FashionMNIST dataset while still providing good performance.

The selection of different architectures for CIFAR-10 and FashionMNIST serves a dual purpose in our study. Firstly, it reflects the inherent complexity of each dataset: the deeper and more sophisticated ResNet18 model is well-suited for CIFAR-10's rich and diverse natural images, while the simpler CNN architecture effectively classifies FashionMNIST's grayscale clothing items. More importantly, this deliberate variation in both datasets and model architectures allows us to investigate the impact of hyper-parameter effects in FL. By examining how hyper-parameters behave across these diverse scenarios, we aim to uncover insights that are robust across different data complexities and model architectures. This approach enhances the broader applicability of our findings and can provides a more in-depth understanding of hyper-parameter dynamics in varied FL contexts.

B. OVERVIEW OF EXPERIMENTS

We conducted four distinct set of experimentation's on each dataset to assess the impact of the hyper-parameters holistically. First, we tuned the learning rate across values of 0.3, 0.1, 0.01, 0.001 and 0.0001 while fixing all other variables. This allowed us to isolate and analyze the effects of varying learning rates. The chosen range of learning rate values covers a broad spectrum, from relatively large values (0.3) to small values (0.0001). This comprehensive range increases the likelihood of capturing the optimal learning rate or at least a suitable range for further fine-tuning. Second, we adjusted the epochs per round to 1, 2, 4, 8, and 16 while controlling other settings to examine the specific influence of this factor. The selected range of values of epochs per round covers a wide spectrum, from minimal local training (1 epoch) to more extensive local training (16 epochs). Our aim was to check the influence of low, medium, and high epoch values, so we chose the range from low to high by powers of 2. This comprehensive range increases the likelihood of capturing the optimal or near-optimal value for the given problem and setup. It will also enables the us to observe potential patterns or trade-offs that may emerge as the number of epochs per round increases or decreases.

Third, we manipulated the batch size from 8 to 128 in factors of 2 to reveal insights on the batch size-performance interplay. The range of batch sizes selected (8, 16, 32, 64, 128) are the most commonly used batch sizes in the literature, and we wanted to see the effects of lower, intermediate, and higher batch sizes on the training dynamics and model performance. This range also enabled us to observe potential patterns or trade-offs that may emerge as the batch size increases or decreases. Finally, we explored CPR values of 5%, 10%, and 25% to construct CPR-efficiency trade-off curves. The range of CPR values selected is among the most common values used in the literature, and usually in FL, a high CPR is avoided due to communication and computational constraints. By systematically varying the CPR across this range while keeping other hyper-parameters constant, we can isolate and study the specific influence of CPR on the FL model's behavior and performance.

Our serialized experiments allow us to identify the independent effect of each hyper-parameter by carefully adjusting one hyper-parameter at a time while maintaining consistency of the other variables for each dataset. For our experiments, we systematically explore a spectrum of values for each hyper-parameter in our experiments as discussed earlier, the default values were set at 0.001, 4, 32, and 5% for learning rate, epochs per round, batch size, and CPR respectively. The subsequent sections of this study show the results obtained from these experiments, their implications and exploration directions in the domain of FL and hyper-parameter optimization.

V. RESULTS AND ANALYSIS

We provided a theoretical overview of the critical hyper-parameters connected with FL and explained their

significance in Section III. Building upon this foundation, this section shows the in-depth analysis of the practical implications and effects associated with each hyper-parameter within the context of a FL setup for each dataset. In our upcoming discussion, we will explore how different settings for hyper-parameters impact how well a FL system works. This analysis goes beyond just theory and we will also discuss what happens in real-world situations when we change the values of these hyper-parameters. Specifically, we will analyze the effects of hyper-parameters such as learning rate, batch size, CPR, and the number of epochs per round. By empirically evaluating these hyper-parameters, we aim to understand their complex relationship and determine their influence on critical aspects of FL, like model loss and training time etc. This detailed examination of hyper-parameter effects in FL aims to provide practical insights that go beyond theory. Our goal is to enhance our understanding of the complexities involved in optimizing FL models for real-world applications.

A. CIFAR-10 DATASET

This subsection discusses the results obtained from experiments done on CIFAR-10 dataset using Resnet-18.

1) IMPACT OF LEARNING RATE

To illustrate the impact of the learning rate in FL, we conducted a series of experiments similar to that of [44], systematically varying the learning rate while maintaining other hyper-parameters at constant values. The learning rates explored were chosen from the set $\{0.0001, 0.001, 0.01, 0.1, 0.3\}$. The effect on loss is visually represented in Figure 1, where the loss curves across rounds are depicted for each learning rate.

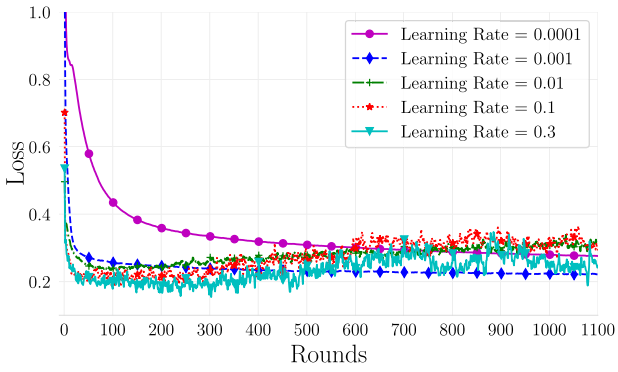

FIGURE 1. Global loss over 1100 rounds with different learning rates on CIFAR-10 dataset.

A notably high learning rate of 0.3 results in highly unstable training, characterized by higher loss and pronounced fluctuations from one round to the next. Beyond 400 rounds, the loss experiences an upward trend, indicating instability. Similarly, a learning rate of 0.1 exhibits suboptimal performance with noticeable oscillations in loss over rounds, indicative of excessive variance and unsuitable for stable convergence. In contrast, a learning rate of 0.01 produces a smoother loss curve with significantly fewer fluctuations, but also loss starts to increase after 100th round. The loss curve for a learning rate of 0.001 stabilizes early and exhibits a gradual reduction over rounds, reflecting a balanced and suitable learning rate. Conversely, the loss curve for a learning rate of 0.0001 is smooth, consistently decreasing, but it is constantly at a higher value than the 0.001 curve, indicating slower convergence. In conclusion, a learning rate of 0.001 emerges as the optimal choice, providing a harmonious blend of stability, convergence speed, and minimal loss among the examined values. Extremely small rates impede training speed, while larger rates destabilize and hinder model convergence. Furthermore, as evidenced by Table 2, learning rate does not significantly influence training time. Regardless of the learning rate values, all trials take nearly the same amount of time to complete 1100 rounds. Consequently, selecting an optimal learning rate in FL becomes crucial for enhancing the overall performance.

2) IMPACT OF NUMBER OF EPOCHS PER ROUND

To elucidate the influence of the number of epochs per round in FL, we conducted experiments with varying epoch sizes across different trials while maintaining consistency in other hyper-parameters. The chosen epoch sizes spanned the set $\{1, 2, 4, 8, 16\}$. The impact of different epochs per round on loss is visually depicted in Figure 2. Utilizing 16 epochs per round resulted in significant overfitting, initially exhibiting a rapid decrease in loss. However, after 200 rounds, the loss began to rise with increasing fluctuations, signaling excessive local iterations without global updates and degradation of convergence. Training with 8 local epochs led to the fastest loss reduction among the tested values, swiftly reaching near 0.2 in about 500 rounds. Nevertheless, the curve displayed a slight increasing trend in later stages, indicating the onset of overfitting and variations in loss throughout the rounds, suggesting incomplete convergence. The loss curve for 4 epochs per round exhibited a decreasing trend

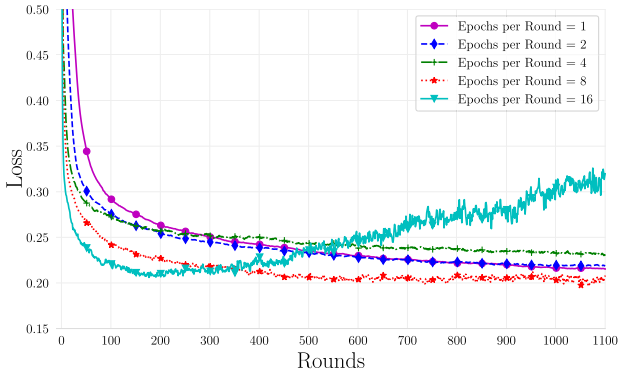

FIGURE 2. Global loss over 1100 rounds with different epochs per round on CIFAR-10 dataset.

throughout the 1100 rounds, with fewer variations indicating a better convergence rate compared to 8 and 16 epochs. Using 2 epochs per round provided more stable convergence, with a smoother loss curve steadily decreasing in later rounds. Remarkably, with only 1 local epoch per round, we observed a smooth loss curve indicating robust convergence, with very minor variations across rounds. Although the convergence at the start was slower compared to other values, it outperformed them at the end of the training.

The number of epochs per round is directly correlated with training time, as evidenced in Table 2. Using fewer epochs per round resulted in less training time, while higher epoch values increased the time required for completion.

In summary, impact of the number of epochs per round in FL is a crucial factor in determining the model's performance and training time. The provided analysis demonstrates that using different numbers of epochs per round can significantly affect the loss and training time of the model. The findings indicate that employing a higher number of epochs per round can lead to overfitting, as evidenced by the loss initially decreasing rapidly but later exhibiting an increasing trend with fluctuations. On the other hand, using a lower number of epochs per round results in reduced training time but may require more rounds to achieve convergence. The analysis also emphasizes the importance of customizing the number of epochs per round for each client to achieve optimal performance in FL.

3) IMPACT OF BATCH SIZE

Similarly, we conducted experiments with different batch sizes while keeping other hyper-parameters fixed. The batch sizes tested were 8, 16, 32, 64, 128. Figure 3 depicts the impact on loss. With a batch of 8, loss reduces rapidly initially, reaching the minimum value of 0.18. However, high fluctuations later signify instability. Increasing to sizes 16, initially shows good performance but the loss started to increase after 500 rounds. With batch sizes of 32 and 64, shows steadier decrease in loss up to 0.22 by round 1100. The almost identical curves indicate marginal performance difference between them. Further increase 128 shows slightly slower convergence versus 32 and 64, achieving slightly higher final loss. No settings overfit within 1100 rounds, as shown by the consistent descent.

Table 2 shows the training times for different batch sizes. We observe that lower batches like 8 take much longer duration in contrast to higher batches, as size increase reduces time. In summary, a batch of 8 starts converging fastest but suffers in stability and efficiency. The 32 and 64 offers better reliability with smooth consistent convergence comparable to higher batches, at intermediate time. Thus, 32 demonstrates the optimal balance of performance and time. Overall, batch size mildly impacts loss trajectory without overfitting, making it useful to tune for efficiency constraints.

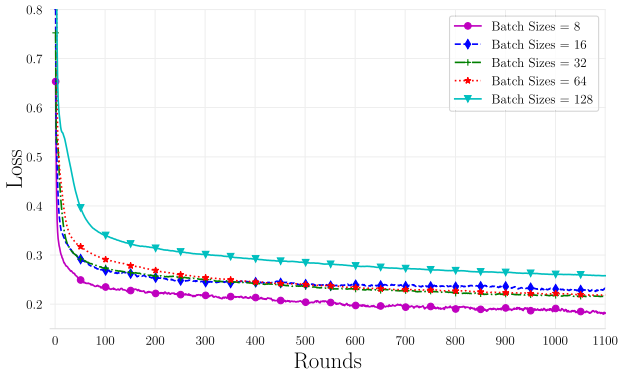

FIGURE 3. Global loss over 1100 with different batch sizes on CIFAR-10 dataset.

4) IMPACT OF CLIENT PARTICIPATION RATIO

Similarly, we evaluated different CPRs between rounds while fixing other hyper-parameters. The CPR percentages tested were 5%, 10% and 25%. Figure 4 visualizes the loss trajectories for each CPR. With 5% CPR, though loss steadily descends, noticeable fluctuations persist across 1100 rounds and the loss is higher in comparison with others. Low participation impairs statistical resiliency. Increasing to 10% CPR markedly smooths out convergence with minimal variations and lowest overall loss. Rapid convergence indicates 10% enables better model updates. However, further increasing CPR to 25% barely improves the loss curve over 10% across rounds. Such diminishing returns imply excessive client updates per round fail to enhance heterogeneity-constrained learning.

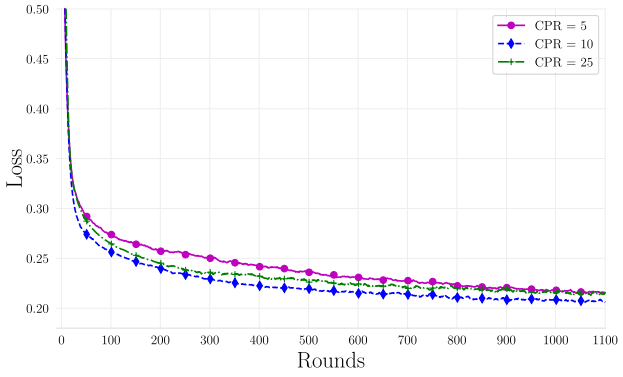

FIGURE 4. Global loss over 1100 with different CPRs on CIFAR-10 dataset.

As per Table 2 shows, training time increases with higher CPR percentages since aggregating grows more expensive. In summary, while 5% CPR demonstrates fluctuating convergence, 10% boosts consistency substantially and minimizes loss. Unfortunately, beyond 10% CPR, loss of consecutive rounds became stagnant, as client diversity starts to impede progress. Therefore, convergence peaks at a moderate 10% CPR before declining with higher rates. Fine-tuning the

percentage of active clients boosts model efficiency and performance.

5) CORRELATION ANALYSIS

The correlation shown in Figure 5 highlights the inter-dependencies between key parameters from our results. It suggests trends such as the impact of epochs on training time, the relationship between batch size and learning rate, and the influence of CPR on various variables. Understanding these correlations is essential for effective hyper-parameter tuning and optimizing the performance of FL models.

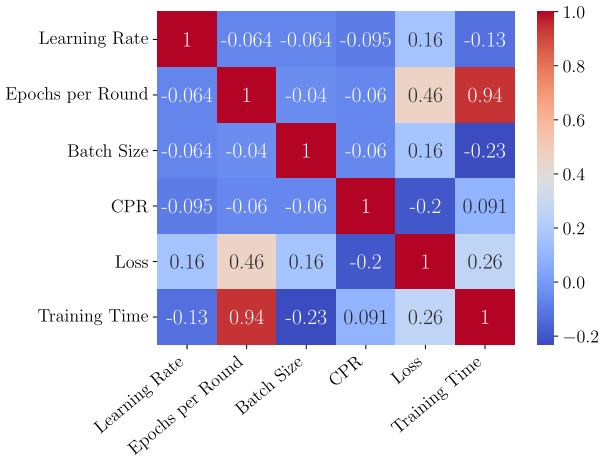

FIGURE 5. Correlation matrix on CIFAR-10 dataset.

The main key take aways from this correlation are as follows:

- There is a moderately positive correlation (0.16) between learning rate and loss - higher learning rates tend to increase loss. But it has very little correlation with training time.
- Epochs per round shows a moderate positive correlation with loss (0.46), indicating that a higher number of epochs is associated with increased loss. Moreover, there is a strong positive correlation (0.94) with training time, suggesting that a increasing the number of epochs leads to a longer training duration.
- Batch size has little correlation with either loss or training time, suggesting tuning it keeps accuracy vs efficiency tradeoff constant. Its interplay is more complex and data-dependent.
- Higher CPR reduces loss moderately (-0.2 correlation) by aggregating more client updates. But it only slightly improves training time. There are diminishing returns for increasing participation.
- Training time exhibits a strong positive correlation with epochs per round (0.94) and a weak positive correlation with loss (0.26). This indicates that longer training times are associated with a higher number of epochs and, to a lesser extent, increased loss.

Moreover, Table 2 provides a detailed summary of results evaluating the isolated and combined effects of learning rate, local epochs, batch size and CPR on loss and training time over 1100 rounds. The key insight is that a learning rate of 0.001, 4 local epochs per round, batch size of 32 and CPR of 10% provides the optimal balance between achieving smooth, fast and stable convergence to low loss while keeping training time reasonable in case of CIFAR10 dataset. Specifically, the experiments quantify how adjusting each hyper-parameter introduces specific trade-offs - larger batches improve accuracy but reduce efficiency, more epochs benefits loss at exponentially higher duration's, larger learning rates destabilize learning while smaller ones slow it down. The interplay between factors is elucidated through the correlational trends. Fundamentally, this table offers empirical evidence to mathematically tune configurations for objectives around speed, cost and accuracy in FL.

TABLE 2. Key hyper-parameter dynamics in federated learning: Insights from CIFAR-10.

Learning Rate	Epochs per Round	Batch Size	CPR	Loss	Training Time(s)
0.0001				0.2759	17343.756
0.001				0.2211	17283.671
0.01	4	32	5	0.3182	17317.275
0.1				0.2934	17261.896
0.3				0.2435	17306.428
	1			0.2152	10373.723
	2			0.2185	13828.978
0.001	4	32	5	0.2299	20731.817
	8			0.2081	33107.668
	16			0.3197	53489.864
		8		0.1818	28035.648
		16		0.2319	21033.067
0.001	4	32	5	0.2155	17287.607
		64		0.2184	15621.533
		128		0.2579	14632.241
			5	0.2155	17287.607
0.001	4	32	10	0.2059	22455.315
			25	0.2138	24258.282

The results suggest that finding the optimal hyper-parameter settings can significantly improve the performance of a FL system. Overall, the correlation provide valuable insights into the impact of hyper-parameters on the performance of a FL system and the need for customized hyper-parameter optimization.

In overall summary, these evaluations performed on CIFAR10 dataset and Resnet-18 dive deeply into the complex dynamics of critical hyper-parameters in this specific case. Across meticulously designed experiments, the study uncovers minute insights, emphasizing the pivotal role of learning rate, epochs per round, batch size, and CPR in optimizing the model performance. The optimal learning rate is identified as 0.001, showcasing a balanced convergence speed and minimal loss. Notably, the number of epochs per round correlates strongly with training time, highlighting its direct impact on the efficiency of FL systems.

Analysis of different batch sizes shows that a batch size of 32 strikes a balance between model efficiency and training time. Additional experiments with different CPR values demonstrate significantly fewer gains above 10%. The correlation analysis provides a deeper understanding of of hyper-parameter inter-dependencies, guiding practitioners in informed decision-making. These CIFAR-10-specific findings underscore the importance of tailored hyper-parameter optimization in FL for complex image classification tasks. The optimal configuration identified can provides a valuable starting point for practitioners working with similar datasets in FL environments. However, it's crucial to note that these results are specific to CIFAR-10 with Resnet-18, and may not generalize directly to other datasets or scenarios.

B. FashionMNIST DATASET

This subsection discusses the results obtained from experiments done on FashionMNIST dataset using a simpler CNN model.

1) IMPACT OF LEARNING RATE

The learning rates explored were chosen from the set {0.0001, 0.001, 0.01, 0.1, 0.3}. The effect on loss is visually represented in Figure 6, where the loss curves across rounds are depicted for each learning rate.

A learning rate of 0.1 shows the fastest convergence and lowest overall loss, reaching a minimum around 0.03 by the end of training. Learning rates of 0.01 and 0.3 also perform well, converging to slightly higher loss values. In contrast, lower learning rates (0.001 and 0.0001) result in slower convergence and higher final loss values. Unlike CIFAR-10, FashionMNIST seems to benefit from higher learning rates, likely due to its simpler nature. Furthermore, as evidenced by Table 3, learning rate does not significantly influence training time. Regardless of the learning rate values, all trials take nearly the same amount of time to complete 1100 rounds.

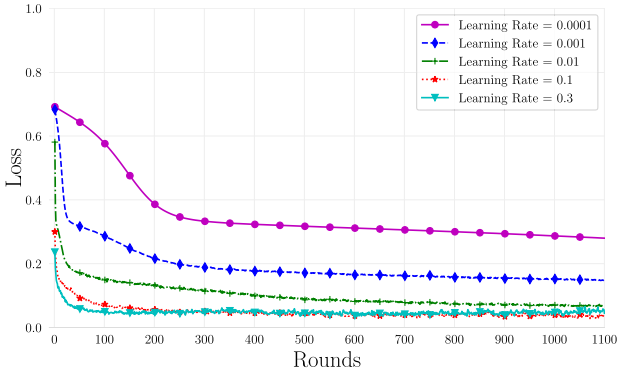

FIGURE 6. Global loss over 1100 rounds with different learning rates on FashionMNIST dataset.

2) IMPACT OF NUMBER OF EPOCHS PER ROUND

The impact of different epochs per round on loss is visually depicted in Figure 7. Using 16 epochs per round leads to the fastest initial decrease in loss and lowest overall loss. As the number of epochs decreases, the convergence becomes slower and final loss values increase. This suggests that for FashionMNIST, more local computation (higher epochs per round) is beneficial, unlike CIFAR-10 where it led to overfitting.

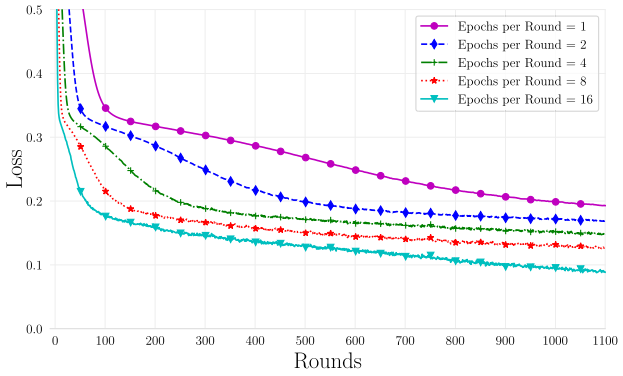

FIGURE 7. Global loss over 1100 rounds with different epochs per round on FashionMNIST dataset.

The number of epochs per round is directly correlated with training time, as evidenced in Table 3. Using fewer epochs per round resulted in less training time, while higher epoch values increased the time required for completion.

3) IMPACT OF BATCH SIZE

Figure 8 illustrates that smaller batch sizes (8 and 16) demonstrate faster initial convergence and achieve lower overall loss. As the batch size increases, we observe a slower convergence rate and higher final loss values. Notably, the batch size of 128 performs significantly worse than the others. These findings suggest that for FashionMNIST, smaller batch sizes are more effective, possibly due to the dataset's relatively simpler nature allowing for more frequent model updates.

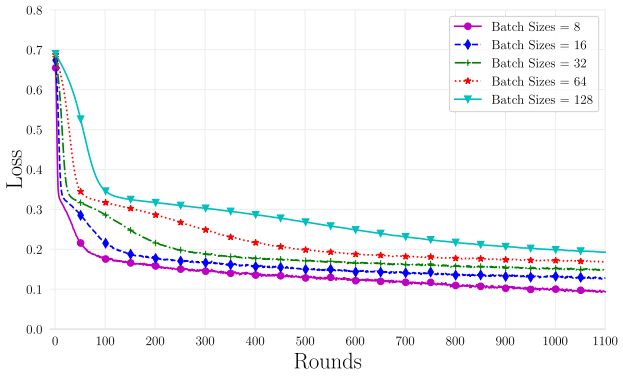

FIGURE 8. Global loss over 1100 with different batch sizes on FashionMNIST dataset.

Table 3 presents the training times for various batch sizes. We note an inverse relationship between batch size and

training duration. Smaller batch sizes, such as 8, require considerably longer training times, while larger batch sizes progressively reduce the overall training duration. This trade-off between convergence speed and training time highlights the importance of carefully selecting batch sizes in FL implementations.

4) IMPACT OF CLIENT PARTICIPATION RATIO

Figure 9 visualizes the loss trajectories for each CPR. Interestingly, all three CPR values (5%, 10%, and 25%) show very similar performance, with nearly identical loss curves. This suggests that for a simpler CNN model and dataset, the model can achieve good performance even with lower client participation, which could be beneficial for reducing communication costs in practical implementations. As per Table 3 shows, training time increases with higher CPR percentages since aggregating grows more expensive.

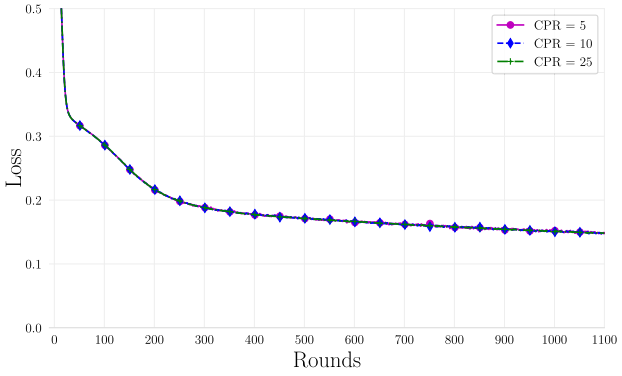

FIGURE 9. Global loss over 1100 with different CPRs on FashionMNIST dataset.

5) CORRELATION ANALYSIS

The correlation shown in Figure 10 highlights the inter-dependencies between key parameters from our results obtained on FashionMNIST dataset and SimpleCNN model.

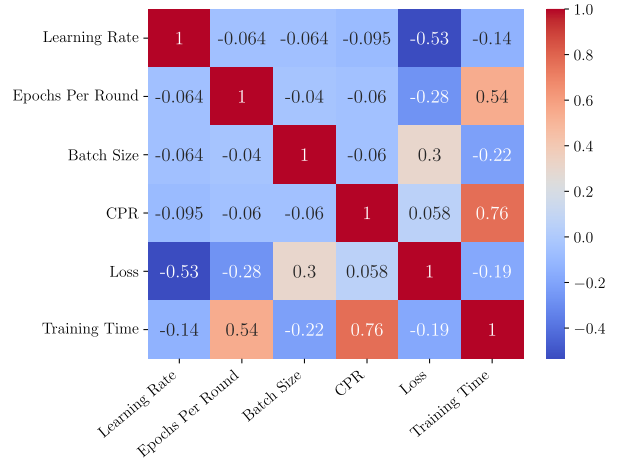

FIGURE 10. Correlation matrix on FashionMNIST.

The main takeaways from this correlation are as follows:

- Learning rate shows a strong negative correlation (−0.53) with loss, indicating that higher learning rates generally lead to lower loss for FashionMNIST. But it has very little correlation with training time.
- Epochs per round has a moderate negative correlation (−0.28) with loss, suggesting more epochs tend to reduce loss. Moreover, there is a strong positive correlation (0.54) with training time, suggesting that a increasing the number of epochs leads to a longer training duration.
- Batch size shows a moderate positive correlation (0.3) with loss, implying larger batch sizes tend to increase loss.
- CPR has a very weak positive correlation (0.058) with loss, confirming the observation from Figure 9 that CPR has little impact on performance for this dataset.
- Training time is strongly positively correlated (0.76) with CPR and moderately positively correlated (0.54) with epochs per round, indicating these factors significantly influence training duration.

Furthermore, Table 3 presents a detailed summary of the results, assessing the individual and combined impacts of learning rate, local epochs, batch size, and CPR on loss and training time across 1100 rounds. A learning rate of 0.1 achieves the lowest loss (0.0321) while maintaining similar training time to other rates. Increasing epochs per round consistently reduces loss, with 16 epochs achieving the lowest loss (0.0886). However, this comes at the cost of significantly increased training time (11446.18 seconds compared to 2200.625 seconds for 1 epoch). Smaller batch

TABLE 3. Key hyper-parameter dynamics in federated learning: Insights from FashionMNIST.

Learning Rate	Epochs per Round	Batch Size	CPR	Loss	Training Time(s)
0.0001				0.2799	4070.287
0.001				0.1478	4069.109
0.01	4	32	5	0.0663	4059.825
0.1				0.0321	4063.516
0.3				0.0447	4060.897
	1			0.1929	2200.625
	2			0.1688	2829.011
0.001	4	32	5	0.1478	4076.377
	8			0.1255	6525.372
	16			0.0886	11446.18
		8		0.0934	7607.835
		16		0.1265	5147.24
0.001	4	32	5	0.1478	4060.765
		64		0.1688	3580.051
		128		0.1929	3390.497
			5	0.1478	4061.776
0.001	4	32	10	0.1482	6769.213
			25	0.1482	14984.292

sizes achieve lower loss, with a batch size of 8 reaching 0.0934 loss. However, smaller batch sizes also increase training time. Interestingly, all tested CPR values (5%, 10%, and 25%) achieve very similar loss (around 0.148), but higher CPR values significantly increase training time.

In summary, for the FashionMNIST dataset and simple CNN model, higher learning rates, more epochs per round, and smaller batch sizes tend to improve performance. The impact of CPR is minimal on loss but significant on training time. These findings contrast with CIFAR-10 in several aspects, highlighting the importance of dataset-specific hyper-parameter optimization in federated learning.

C. COMPARATIVE ANALYSIS: CIFAR-10 VS FashionMNIST

The study reveals significant differences in hyper-parameter dynamics between FashionMNIST and CIFAR-10 datasets in FL, highlighting the impact of dataset complexity and model architecture on optimal hyper-parameters.

FashionMNIST, a simpler dataset paired with a basic CNN model, demonstrated higher tolerance for aggressive hyper-parameters. It showed improved performance with higher learning rates, such as 0.1, which achieved the lowest loss and fastest convergence. The correlation analysis indicates a positive relationship between higher learning rates and improved performance metrics for FashionMNIST. Similarly, increasing the number of epochs per round (up to 16) led to faster convergence and lower overall loss, suggesting that more local computation before global updates is beneficial for simpler datasets.

In contrast, the more complex CIFAR-10 dataset, using ResNet-18, required more conservative settings to avoid instability and ensure stable convergence. A moderate learning rate of 0.001 was necessary to maintain stability and achieve optimal performance, with higher learning rates showing a negative correlation with performance metrics due to increased instability and loss. For CIFAR-10, increasing epochs per round beyond 4 led to overfitting and instability, as shown by a negative correlation with performance metrics at higher values.

These findings imply that simpler datasets can benefit from more agressive learning rates and epochs per round, without risking instability, while complex datasets require more careful tuning to avoid overfitting and ensure stable convergence.

Interestingly, both datasets exhibit similar patterns when it comes to batch size variations. Smaller batch sizes generally resulted in lower training loss but significantly increased the training time. This trade-off indicates that while smaller batches may enhance model accuracy by providing more granular updates, they require considerably more time to converge due to the higher number of updates needed. On the other hand, larger batch sizes tended to reduce training time but at the cost of higher training loss. Notably, a batch size of 32 emerged as the optimal choice for both CIFAR-10 and FashionMNIST, striking an effective balance between minimizing training loss and maintaining a reasonable training time. This batch size allowed for efficient training while preserving the accuracy of the model.

The study also reveals significant differences in how CPR affects performance. For CIFAR-10 using ResNet-18, the optimal CPR was around 10%, with diminishing returns for higher values. FashionMNIST, using a simpler CNN model, showed minimal loss differences across CPR values but increased training time for higher CPRs. Notably, FashionMNIST achieved good performance even with lower CPR, suggesting simpler datasets can maintain effectiveness with reduced client participation.

These insights underscore the critical importance of dataset-specific hyper-parameter optimization in FL, balancing performance gains against practical constraints. While simpler datasets like FashionMNIST can benefit from more aggressive settings, such as higher learning rates and more epochs per round, they may not always require such stringent optimization, offering greater flexibility in system design. In contrast, complex datasets like CIFAR-10 demand more conservative settings to avoid instability and overfitting. These differences highlight the need for hyper-parameter optimization tailored to the specific dataset and task. Without this tailored approach, there is a risk of suboptimal performance, inefficient training, and potential instability, especially in more complex tasks. Therefore, achieving optimal performance in FL requires careful optimization of hyper-parameters, considering the unique characteristics and complexities of each dataset, model architecture, and task at hand.

VI. CONCLUSION AND FUTURE SCOPE

In conclusion, this study elucidates the complex relationships between hyper-parameters and FL model performance. Through extensive experiments on CIFAR-10 and FashionMNIST datasets, we have highlighted the importance of dataset-specific hyper-parameter optimization. Our experiments on CIFAR-10 and FashionMNIST reveal that optimal hyper-parameter configurations can significantly vary depending on the dataset characteristics and model complexity. The correlation analysis conducted further enriches our understanding of the inter-dependencies between these parameters, aiding practitioners in making informed decisions during hyper-parameter optimization.

The differences in results between CIFAR-10 and FashionMNIST underscore the necessity of dataset-specific hyper-parameter optimization in FL. While certain hyper-parameters like learning rate and batch size universally affect model performance, their optimal values and impacts can significantly differ based on the dataset and model used. This variability necessitates a tailored approach to hyper-parameter optimization for each specific task to ensure optimal performance and efficiency. These findings contribute significantly to the understanding of hyper-parameter dynamics in FL, offering practical guidance for practitioners in optimizing FL systems across varied applications and data complexities.

For future research, exploring adaptive strategies for hyper-parameter optimization that dynamically adjust to the evolving characteristics of participating clients could be a promising avenue. Addressing the challenges posed by non-IID data distributions and heterogeneous network conditions remains a crucial aspect of advancing FL research. Overall, this study sets the stage for further advancements in hyper-parameter optimization for FL, contributing to the ongoing evolution of privacy-preserving and decentralized ML paradigms.

REFERENCES

[1] B. Rao, J. Zhang, D. Wu, C. Zhu, X. Sun, and B. Chen, "Privacy inference attack and defense in centralized and federated learning: A comprehensive survey," *IEEE Trans. Artif. Intell.*, early access, Feb. 8, 2024, doi: 10.1109/TAI.2024.3363670.

[2] J. Koneč ný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[3] P. K. Quan, M. Kundroo, and T. Kim, "Experimental evaluation and analysis of federated learning in edge computing environments," *IEEE Access*, vol. 11, pp. 33628–33639, 2023.

[4] J. Lu, N. Fukumoto, and A. Nakao, "A security-oriented overview of federated learning utilizing layered reference model," *IEEE Access*, vol. 12, pp. 103949–103975, 2024.

[5] A. Padma and M. Ramaiah, "Blockchain based an efficient and secure privacy preserved framework for smart cities," *IEEE Access*, vol. 12, pp. 21985–22002, 2024.

[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[8] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-IID data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24462–24474, 2021.

[9] S. Zawad and F. Yan, "Hyperparameter tuning for federated learning—Systems and practices," in *Federated Learning*. Amsterdam, The Netherlands: Elsevier, 2024, pp. 219–235.

[10] N. Mitic, A. Pyrgelis, and S. Sav, "How to privately tune hyperparameters in federated learning? Insights from a benchmark study," 2024, *arXiv:2402.16087*.

[11] M. Khodak, R. Tu, T. Li, L. Li, M.-F. F. Balcan, V. Smith, and A. Talwalkar, "Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 19184–19197.

[12] M. Kundroo and T. Kim, "Federated learning with hyper-parameter optimization," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 9, Oct. 2023, Art. no. 101740. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S131915782300294X

[13] Y. Zhou, P. Ram, T. Salonidis, N. Baracaldo, H. Samulowitz, and H. Ludwig, "FLoRA: Single-shot hyper-parameter optimization for federated learning," 2021, *arXiv:2112.08524*.

[14] J. Parra-Ullauri, X. Zhang, A. Bravalheri, R. Nejabati, and D. Simeonidou, "Federated hyperparameter optimisation with flower and optuna," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, 2023, pp. 1209–1216.

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

[16] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," 2020, *arXiv:2003.00295*.

[17] X. Ma, R. Bao, J. Jiang, Y. Liu, A. Jiang, J. Yan, X. Liu, and Z. Pan, "FedSSO: A federated server-side second-order optimization algorithm," 2022, *arXiv:2206.09576*.

[18] A. Koskela and A. Honkela, "Learning rate adaptation for federated and differentially private learning," 2018, *arXiv:1809.03832*.

[19] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, Eds., Cham, Switzerland: Springer, 2019, pp. 92–104.

[20] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for (parallel) stochastic optimization," in *Proc. IEEE 51st IEEE Conf. Decis. Control (CDC)*, Dec. 2012, pp. 5442–5444.

[21] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 9793–9803. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file /90365351ccc7437a1309dc64e4db32a3-Paper.pdf

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[23] S.-I. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, nos. 4–5, pp. 185–196, Jun. 1993.

[24] D. Mistry, M. F. Mridha, M. Safran, S. Alfarhood, A. K. Saha, and D. Che, "Privacy-preserving on-screen activity tracking and classification in e-learning using federated learning," *IEEE Access*, vol. 11, pp. 79315–79329, 2023.

[25] W. Liu, X. Zhang, J. Duan, C. Joe-Wong, Z. Zhou, and X. Chen, "AdaCoOpt: Leverage the interplay of batch size and aggregation frequency for federated learning," in *Proc. IEEE/ACM 31st Int. Symp. Quality Service (IWQoS)*, Jun. 2023, pp. 1–10.

[26] H. Mostafa, "Robust federated learning through representation matching and adaptive hyper-parameters," 2019, *arXiv:1912.13075*.

[27] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[28] K. Murphy, *Machine Learning: A Probabilistic Perspective* (Adaptive Computation and Machine Learning Series). Cambridge, MA, USA: MIT Press, 2012. [Online]. Available: https://books.google.co.kr/books?id=NZP6AQAAQBAJ

[29] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[30] D. Passos and P. Mishra, "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks," *Chemometric Intell. Lab. Syst.*, vol. 223, Apr. 2022, Art. no. 104520.

[31] M. Dib, B. Ribeiro, and P. Prates, "Federated learning as a privacy-providing machine learning for defect predictions in smart manufacturing," *Smart Sustain. Manuf. Syst. ASTM J.*, vol. 5, no. 1, pp. 1–18, 2021.

[32] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," 2017, *arXiv:1711.00489*.

[33] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, D. Deng, and M. Lindauer, "Hyperparameter optimization: Foundations, algorithms, best practices and open challenges," 2021, *arXiv:2107.05847*.

[34] Z. Charles, Z. Garrett, Z. Huo, S. Shmulyian, and V. Smith, "On large-cohort training for federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 20461–20475.

[35] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 37–53, Jan. 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:238059715

[36] H. Zhang, M. Zhang, X. Liu, P. Mohapatra, and M. DeLucia, "FedTune: Automatic tuning of federated learning hyper-parameters from system perspective," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2022, pp. 478–483.

[37] K. Kuo, P. Thaker, M. Khodak, J. Nguyen, D. Jiang, A. Talwalkar, and V. Smith, "On noisy evaluation in federated hyperparameter tuning," in *Proc. Mach. Learn. Syst.*, vol. 5, 2023, pp. 1–18.

[38] B. Alhalabi, S. Basurra, and M. M. Gaber, "FedNets: Federated learning on edge devices using ensembles of pruned deep neural networks," *IEEE Access*, vol. 11, pp. 30726–30738, 2023.

[39] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674862X19300047

[40] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[41] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[42] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[44] J. Zhang, Y. Zhao, F. Shone, Z. Li, A. F. Frangi, S. Q. Xie, and Z.-Q. Zhang, "Physics-informed deep learning for musculoskeletal modeling: Predicting muscle forces and joint kinematics from surface EMG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 484–493, 2023.

MAJID KUNDROO (Graduate Student Member, IEEE) received the M.S. degree in computer science from the Islamic University of Science and Technology, Jammu and Kashmir, India, in 2019. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Chungbuk National University, South Korea. His research interests include edge computing, edge AI, the Internet of Things, and federated learning.

TAEHONG KIM (Senior Member, IEEE) received the B.S. degree in computer science from Ajou University, Republic of Korea, in 2005, and the M.S. degree in information and communication engineering and the Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), in 2007 and 2012, respectively. He worked as a Research Staff Member at the Samsung Advanced Institute of Technology (SAIT) and the Samsung DMC Research and Development Center, from 2012 to 2014. He also worked as a Senior Researcher at the Electronics and Telecommunications Research Institute (ETRI), from 2014 to 2016. Since 2016, he has been an Associate Professor with the School of Information and Communication Engineering, Chungbuk National University, Republic of Korea. His research interests include edge computing, container orchestration, the Internet of Things, and federated learning. He has been an Associate Editor of IEEE Access, since 2020.