DEMYSTIFYING HYPERPARAMETER OPTIMIZATION IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) is a new machine learning paradigm that enables training models collaboratively across clients without sharing private data. In FL, data is non-uniformly distributed among clients (i.e., data heterogeneity) and cannot be balanced nor monitored like in conventional ML. Such data heterogeneity and privacy requirements bring unique challenges for learning hyperparameter optimization as the training dynamics change across clients even within the same training round and they are difficult to measure due to privacy constraints. State-of-the-art frameworks in FL focus on developing better aggregation algorithms and policies with the aim of mitigating these challenges. However, almost all existing FL systems adopt a "global" tuning method that uses a single set of learning hyperparameters across all the clients, regardless of their underlying data distributions. Our study shows that such a widely adopted global tuning method is not suitable for FL due to its data heterogeneity-oblivious nature. We demonstrate that the data quantity and distribution of the clients have a significant impact on the choice of hyperparameters, making it necessary to have customized tuning for each client. Based on these observations, we propose a first of its kind privacy preserving and heterogeneity-aware hyperparameter optimization methodology, FedTune, that adopts a proxy data based hyperparameter customization approach to address the privacy and tuning cost challenges. Together with a Bayesian strengthened tuner, the proposed customized tuning approach is effective, lightweight, and privacy preserving. Extensive evaluation demonstrates that FedTune can achieve up to 7/4/4/6% better accuracy than the widely adopted globally tuned method for popular FL benchmarks FEMNIST, Cifar100, Cifar10, and Fashion-MNIST respectively.

1 INTRODUCTION

In conventional distributed learning, data needs to be centrally collected and managed, which is infeasible for scenarios where privacy and security are required. For example, new legislation such as the General Data Protection Regulation (GDPR) (Goddard (2017)) and the Health Insurance Portability and Accountability Act (HIPAA) (O'herrin et al. (2004)) prohibit transferring user private data to a centralized location. Federated Learning (FL) is a new machine learning paradigm that supports collaborative learning with data privacy preserving across data owners (a.k.a, clients) (Konečnỳ et al. (2016); Smith et al. (2017b)). In FL, training is performed at each client and only the trained model weights instead of data are sent from the clients to a centralized server (a.k.a aggregator) for aggregation.

Despite of the support on privacy and security, such data decentralized training paradigm results in data heterogeneity (Li et al. (2020); Nishio & Yonetani (2019)) as clients usually have different amount of data (*data quantity heterogeneity*) and different distribution of data (*data distribution heterogeneity*). Worse, due to the privacy requirement, data cannot be balanced nor monitored. Data heterogeneity is an open problem that is known to cause severe issues on learning performance (Li et al. (2020)). Recent works try to alleviate the data heterogeneity impact by innovating on the aggregation algorithm and client selection policy (Li et al. (2018); Bonawitz et al. (2019); Wang et al. (2020)). While these approaches can reduce the impact of data heterogeneity, data heterogeneity remains a big challenge, e.g., compared to conventional learning on the same benchmark, FL can suffer up to 15% accuracy loss (Chai et al. (2020); Yang et al. (2021); McMahan et al. (2021)).

One important aspect of FL that is largely overlooked by existing works and sometimes considered as mystery is the hyperparameter optimization. Existing FL systems simply apply the same approach of conventional distributed learning for hyperparameter optimization, i.e., tuning one set of hyperparameters via trial and error and applying them to all clients indistinguishably. Our study shows that such data heterogeneity-oblivious approach may lead to severe performance drops. This is because data heterogeneity can cause quite different learning dynamics across clients and such one-size-fits-all solution often strikes a poor balance. As an example, if we have two clients, Client A with 64k data samples and Client B with 64 data samples, and a global mini-batch size of 64, it would result in too many model parameter updates (i.e., 1000) for Client A while too few updates (i.e., 1) for Client B. Such imbalanced learning process among clients would ultimately result in poor model performance.

To demystify this complex interdependency between data heterogeneity and hyperparameters, we conduct an in-depth study to understand the uniqueness and challenges of hyperparameter optimizations in FL. From the extensive experimental results, we observe the following opportunities and challenges: 1) Heterogeneity-aware hyperparameter optimization, i.e., adaptively customized hyperparameters for each client, based on data heterogeneity, can significantly improve the overall model performance compared to existing heterogeneity-oblivious approaches. 2) Hyperparameter customization improves accuracy but imposes nontrivial hyperparameter optimization overhead. Since FL often involves a huge number of clients, customizing hyperparameters for each individual one would lead to tremendous cost. 3) Hyperparameter customization incurs additional technical challenges, as data and training dynamics cannot be shared nor monitored due to privacy constraints.

Based on the above key findings, we propose *FedTune*, a first of its kind privacy preserving and heterogeneity-aware hyperparameter optimization methodology. To solve the privacy preserving challenge and reduce the tuning overhead on the client side, *FedTune* introduces a novel sampled proxy-based hyperparameter customization method by tuning the hyperparameters on this dataset on the aggregator side and only sending the populated Hyperparameter Reference Table to clients to make hyperparameter choice decisions. This sampled proxy dataset requires no information about client data and thus preserves client privacy. In addition, it incurs no additional overhead on the client side. To further reduce the tuning overheads on the aggregator side, *FedTune* employs a tuning approach strengthened by Bayesian Optimization, which significantly accelerates the hyperparameter customization speed.

We prototype *FedTune* on a real distributed FL testbed and evaluate its effectiveness and robustness using popular and sophisticated FL benchmarks. Extensive evaluation demonstrates the superior performance advantage of *FedTune* over the widely used heterogeneity-oblivious method, with reduced tuning overheads compared to Random Search and Grid Search. For FEMNIST of LEAF (Caldas et al. (2018)), the most sophisticated available FL benchmark, we achieve an accuracy improvement of up to 7%.

2 Background and Related Work

2.1 Data Heterogeneity in Federated Learning

FL often involves a large number of clients such as mobile or IoT devices with limited computing capacity and unreliable communication. In each training round, only a small portion of clients is selected to participate. Each participated client trains the latest model weights locally and only sends the trained weights/gradients in a secured manner (Shokri & Shmatikov (2015); Bonawitz et al. (2017)) to the central aggregator to produce the new aggregated weights. During the entire training process, client's data is not shared with other clients nor the aggregator, thus preserving the privacy.

Data heterogeneity is an important feature of FL as clients usually have different amounts and distribution of data (Ghosh et al. (2019); Li et al. (2019)). Although data heterogeneity may also exist in conventional machine learning, the extent of data heterogeneity is much more pronounced in FL (Yang et al. (2021); Li et al. (2019)). In addition, unlike conventional machine learning, data in FL cannot be balanced nor monitored due to the privacy requirements. As data heterogeneity has severe impacts on learning performance, there are several lines of works that try to mitigate the data heterogeneity impact.

The first line of works focus on designing a client selection policy for choosing devices with similar data distribution. FAVOR (Wang et al. (2020)) is designed to reduce the bias and heterogeneity in the system by carefully choosing a subset of clients using Reinforcement Learning. Qian et al. (2020)

proposes a new method that mathematically and empirically sample clients to adjust the heterogeneity level from clients' update. TiFL (Chai et al. (2020)) employs a tiered strategy to lower the data heterogeneity in each training round. Although these works mitigate the data heterogeneity impact, they have two major limitations. First, these methods require to know the client data distributions, which can violate strict privacy constraints. In addition, they may lead to biased sampling of clients for training that may cause "fairness" issues Mohri et al. (2019).

Another line of works propose strategies to identify which parts of the model are mostly affected by data heterogeneity, and then apply regularization methods (such as weight regularization) to mitigate the heterogeneity data impact. One notable example is using meta-learning to find an initial shared model that can be adapted to the heterogeneous data of clients (Fallah et al. (2020)). However, their focus is mostly on ensuring a high test performance on the client's individual datasets (i.e. better personalized local models) instead of finding a generalizable model which performs well on an unbiased dataset. Similarly, FedEX (Khodak et al. (2020)) proposes the use of neural architecture search (NAS) and weight-sharing techniques to make the local models more personalized. Another method is SCAFFOLD (Karimireddy et al. (2020)), where noise is applied to the local weights before being sent over to the aggregator. However, these solutions only work under a certain bound of data heterogeneity.

The most relevant line of works innovate on the aggregation algorithms to mitigate the data heterogeneity impact in FL. In Reddi et al. (2020), three algorithms are proposed based on FedAvg, namely FedAdaGrad, FedYogi, and FedAdam. FedDF (Lin et al. (2020)) allows flexible aggregation over heterogeneous client models by using ensemble distillation (Hinton et al. (2015)0. Pillutla et al. (2019) uses median and variance instead of averaging in the aggregation algorithm to mitigate the impact of highly diverse weights between clients. Yao et al. (2019) trains the global model after aggregation on a small balanced dataset on the server-side to alleviate the biased local training. Zheng et al. (2021); Wu et al. (2020b) use hierarchical aggregation servers which averages clients with similar data distributions to achieve a more stable training process. While these works demonstrate that it is possible to mitigate the data heterogeneity problem from the server side, our hyperparameter optimization methodology is complimentary to them. We demonstrate in our evaluation that together with our methodology, the performance can be further improved.

2.2 Hyperparameter Optimization

The performance of machine learning models are sensitive to hyperparameters. Hyperparameter optimization (HPO) aims at tuning the hyperparameters to improve convergence speed and quality. However, due to the wide range of hyperparameter choices and their corresponding dynamic schedules, tuning these hyperparameters is a time and resource consuming task. To improve the efficiency of hyperparameter optimization, various works have been proposed, including multi-armed bandits, evolutionary algorithms, and Bayesian Optimization (BO). Hyperband (Li et al. (2017)) treats the problem of hyperparameter tuning as a many-armed bandit, and discards the worst configurations during different tuning stages. (Young et al. (2015)) introduces the framework MENNDL for optimizing hyperparameters using genetic algorithms. A new BO method was introduced in Wu et al. (2020a) by creating its acquisition function to leverage multi-fidelity feature. However, all the existing HPO works are designed specifically for conventional machine learning. To the best of our knowledge, there is no customized hyperparameter optimization work addressing the data heterogeneity and privacy issues, which are the key properties of FL.

3 Federated Learning Hyperparameter Optimization Demystified

In this section, we perform an empirical study on hyperparameter optimization of Federated Learning to understand its uniqueness, opportunities, and challenges.

3.1 Why Hyperparameter Optimization in Federated Learning is So Unique?

A Different Training Procedure Conventional machine learning trains a model by iterating over the entire training data in epochs. In each training round, all training nodes perform forward-passes on mini-batches and send their weights/gradients to the server to perform optimization such as Stochastic Gradient Descent (SGD). However, in FL only a small fraction of clients participate in the training each round. The participating clients complete the full training locally and send the final model weights to the server. The server only performs an aggregation (McMahan et al. (2017)) and the

actual weight optimization is at the individual client level. Such a unique training procedure of FL indicates the conventional "global" hyperparameter tuning may not be suitable for FL as there is no "global" optimization.

Every Client Is Unique. In FL, each client is unique as it has its own private data with different *data distribution* and *data quantity*. The unique training procedure further amplifies the data heterogeneity impact on the learning performance. In addition, the private data of clients cannot be balanced nor monitored, making the data heterogeneity an intrinsic property of FL. To quantify the data heterogeneity, we look into the *data distribution* and *data quantity* respectively.

Data distribution heterogeneity means the data of different client may have different features, which is often due to the unique behaviors of clients in generating or storing data. Take image classification of cats and dogs as an example, cat-owners usually have more cat images than dog images on their phones. Such data distribution heterogeneity may cause performance issues, e.g., a model trained on cat owners may have better performance on cat images than dog images, and vice versa (Li et al. (2019; 2018); Chai et al. (2020); Konečnỳ et al. (2016)). There are different metrics that can be used to quantify data distribution heterogeneity. For our experiments, we use the commonly used Heterogeneity Index (*HI*) (Zawad et al. (2021); Li et al. (2019)) for our setups (more details in Section 5.1). Other distribution metrics such as Gaussian and Poisson are also used with similar results and presented in the Appendix. Heterogeneity index, denoted as $HI(c)$, is defined as a normalized measurement of data distribution heterogeneity: $HI(c) = 1 - \frac{1}{c_{max}-1} \times (c-1), c \in [1, c_{max}]$, where c controls the heterogeneity by adjusting the number of classes per client out of the total number of classes c_{max} in the full dataset. $HI(c)$ ranges from 0 to 1, where 0 represents a completely balanced synthetic dataset and 1 means there are only data points with 1 class on the device, which is the highest level of imbalanced data distribution possible.

Data quantity heterogeneity means that the amount of data may vary from client to client. This is also due to user behaviors. For example, clients who text a lot have more data points to train for a word-prediction model than clients who text very little. Such heterogeneity may also impact performance during the training process.

Privacy Requirements. In FL, the data of a client cannot be shared with other clients nor the server. Even the properties of client data are strictly private and cannot be revealed. This makes the already impactful data heterogeneity even more challenging as the data heterogeneity of a client cannot be measured or inferred. Without knowing the data heterogeneity information, conventional wisdom for mitigating data heterogeneity impact is difficult to be adopted in FL (Li et al. (2020)).

3.2 Understanding Hyperparameter Optimization in Federated Learning

In this section, we conduct an empirical study to understand the opportunities and challenges of hyperparameter optimization in FL. For detailed experimental setup, please refer to Section 5.1.

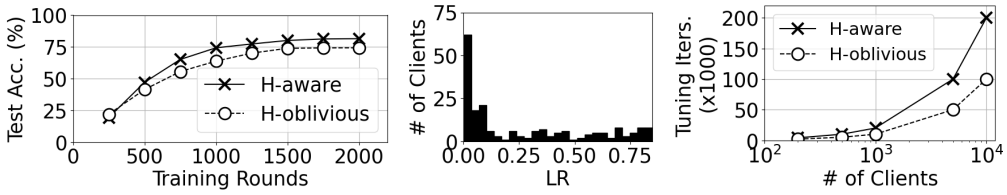
3.2.1 Heterogeneity-oblivious VS Heterogeneity-aware Hyperparameter Opt.

To understand whether data heterogeneity-aware makes a big difference in FL hyperparameter optimization, we compare the performance of the following two different approaches.
Heterogeneity-oblivious (H-oblivious): Like all existing FL works, we hand tune a global set of hyperparameters and use it across all clients.
Heterogeneity-aware (H-aware): We hand tune the hyperparameters for each client to have customized hyperparameters per client to be aware of the data heterogeneity.

In the experiment, we use FEMNIST (Caldas et al. (2018)), a popular image classification benchmark with data heterogeneity (see Sec. 5.1 for setup details). We hand tune the local hyperparameters learning rate, batch size, and local epochs (defined as the number of training epochs at each client) for each client. We follow literature (Caldas et al. (2018)) to set the search range of hyperparameters and apply grid search to identify the best hyperparameters for individual clients. The distribution of the learning rate hyperparameters are given in Figure 1b (batch sizes are given in Appendix Table 2).

The comparison of test accuracy curves across training rounds is shown in Figure 1a. We can see *Heterogeneity-aware* outperforms *Heterogeneity-oblivious* in accuracy most of the time during the training process and yields a 7.4% better accuracy after both training configurations plateau. This is because the one-size-fits-all approach of *Heterogeneity-oblivious* is inevitably unfavorable for some clients no matter how judiciously the tuning is due to the data heterogeneity across clients. On the

other hand, the customization approach of *Heterogeneity-aware* can tailor the hyperparameters for each client to maximize the performance benefits. Figure 1b demonstrates the hand-tuned learning rate distribution across clients. The distribution demonstrates that while a majority of the learning rates are within the $0.001 - 0.1$ range, there are lots of clients requiring a much more diverse range of values, making it costly to tune them individually by hand. These observations lead to the finding – *Data heterogeneity-aware hyperparameter optimization approaches have the potential to improve the learning performance compared to data heterogeneity-oblivious approaches.*



(a) (b) (c)

Figure 1: (a) Test accuracy vs. training rounds comparison. (b) The tuned learning rates and the corresponding number of clients that use them. Derived after hand-tuning all clients. (c) Scalability of the number of training iterations that must be run to tune hyperparameters with varying number of clients in the system.

3.2.2 Challenges of Heterogeneity-aware Hyperparameter Optimization

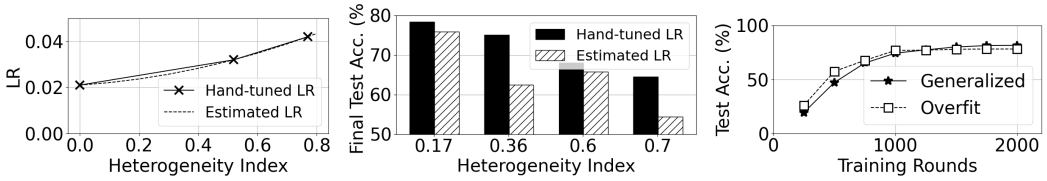
Despite of the potential benefits of data heterogeneity-aware hyperparameter optimization for FL, there are two major challenges for applying this method in practice.

Expensive tuning cost – Heterogeneity-oblivious methods only tune one set of hyperparameters across all clients thus having low tuning costs. However, data heterogeneity-aware methods need to perform customized tuning on each client and the cost accumulates over clients. This is especially significant in FL due to the large number of clients. In addition, clients in FL are usually IoT or mobile devices with limited computing capacity. Performing computing intensive hyperparameter tuning tasks on these devices is slow and may degrade the user experience. In our experiments, we perform 20 iterations of training per client to get an estimate of how well a particular set of hyperparameters can perform when hand-tuning them individually. As we increase the number of clients in the system, the number of tuning iterations increases linearly. For global tuning (i.e. *Heterogeneity-oblivious*), we follow Chai et al. (2020); Qian et al. (2020) to train the full FL system with the currently being explored hyperparameter set for 100 rounds with 10% of total number of clients selected per round, meaning that the total number of tuning iterations when exploring is 10% of the total clients times 100. Figure 1c shows the comparison of tuning cost measured in the number of tuning iterations between *Heterogeneity-oblivious* and *Heterogeneity-aware* methods at different scale, i.e., with different total number of clients. Note that the number of tuning iterations refers to the "total" tuning iterations across all clients, so it equals to *the number of clients times the number of total rounds the client has participated in*. We can see that both approaches have increased tuning cost when the scale of FL increases. However, *Heterogeneity-oblivious* always has much smaller cost compared with *Heterogeneity-aware*, this gap is especially large when the scale is large. This indicate even though *Heterogeneity-aware* approach has better performance benefits, it does not scale well due to the quickly increasing tuning cost. As such, we must opt for a tuning system that preferably offloads the tuning on the server side.

Offloading tuning tasks to server is challenging – Given the expensive tuning cost and poor scalability of *Heterogeneity-aware* approach, one natural question is can we minimize the tuning cost? For example, one of the reasons why the tuning cost is significant is due to the limited computation capacity of clients. If we can offload the tuning to the server, which is usually hosted in the cloud or data centers with plenty of computing resources, the tuning cost concern would be much smaller. Unfortunately, due to the privacy requirements in FL, neither the data nor the properties of data (e.g., data heterogeneity information) of a client can be shared with the server. Without knowing the data or its property, offloading the tuning task to the server becomes quite challenging.

Utilizing the patterns of hyperparameters is challenging – Another natural thought is whether hyperparameters have patterns and can be estimated instead of tuned? To test this out, we run a simple experiment where we use 6 different datasets with data quantity of 400 and 800 combined

with *HI* of 0.15, 0.5, and 0.75 respectively. We set the batch size of 5 and 10 for 400 samples dataset, and 20 and 30 for 800 samples dataset respectively. We hand tune the learning rate to achieve the best performance. The optimal learning rate is shown in Appendix Table 2. We can see the optimal learning rates have a clear pattern – *with the increasing of heterogeneity level while other factors are the same, the learning rate increases.* Also, with more training iterations (the ratio of data set size and batch size), the learning rate also increases. This observation is corroborated by the paper (Smith et al. (2017a)), where they derive the relationship between the noise scale, i.e., the magnitude of the random fluctuations in the training dynamics, and the learning hyperparameters. Specifically, they suggest that the learning rate should increase with increased noise scale. In the federated learning case, the higher the heterogeneity, the noisier the training process (Zhao et al. (2018); Li et al. (2019)) which is why we observe that an increase in data heterogeneity requires higher learning rates.



(a) (b) (c)

Figure 2: (a) Hand-tuned vs. estimated learning rate (LR) under different heterogeneity index. Estimation is done via regression fitting. (b) Final test accuracy comparison between hand-tuned vs. estimated LR under different heterogeneity index. (c) Comparison of test accuracy across training rounds for generalized vs. overfit tuning.

Such pattern seems to be helpful in reducing the tuning overhead as we can estimate the learning rate based on the pattern via a fitting method. Figure 2a shows the pattern of the optimal learning rates for the system's corresponding *HI* (Hand-tuned LR). We fit a quadratic regression model (Estimated LR) and interpolate the LR values for other *HI*. Figure 2b shows the difference of the accuracies derived with the interpolated learning rates against their optimal hand-tuned values. For example, here if we estimate the learning rate at *HI* 0.36 via interpolation, the estimated learning rate is 0.029, very close to the actual optimized learning rate 0.034. However, we observe from Figure 2b that the accuracy when using the estimated learning rate is around 15% lower than the accuracy of tuned learning rate. This study suggests it is challenging to utilize the patterns of hyperparameters for estimating optimal hyperparameter values to reduce the tuning cost since the learning rates are very sensitive. However, this pattern is sufficient such that it can be used by the acquisition functions in BO as a guide during the exploration phase e.g., by avoiding exploring known bad hyperparameters.

Forgetting factor requires a new privacy preserving evaluation method of tuning – Due to the privacy requirements, the evaluation of hyperparameter tuning also becomes a challenging problem. In conventional machine learning, the hyperparameter tuning is evaluated using the global model at the server side. However, in federated learning, client data is strictly private and cannot be used for evaluating the tuned hyperparameters. In addition, due to the data heterogeneity across client data, using client data may lead to overfitting of certain features. To preserve privacy and avoid the overfitting problem due to data heterogeneity, we use an independent and identically distributed (IID) dataset, i.e. synthetic dataset, that *contains no client information for evaluating the hyperparameter tuning.*

Here, we obtain this dataset by sampling from the portion of FEMNIST dataset that is not used for training. For discussions of how to get such data in practice, see Section 4.1. To verify its effectiveness, we compare the test accuracy across training rounds using client data (named Overfit) vs. synthetic dataset (named Generalized) for evaluation in Figure 2c. From the results, we can see at the beginning, using client data achieves better test accuracy due to the overfitting, but using the synthetic dataset eventually achieves better test accuracy. This is because of the forgetting factor in FL – *overfitting enforces the models to favor trained features temporally, but when models are trained with new features overtime, the earlier trained features are easier to be forgotten,* a unique phenomenon in FL (Sun et al. (2019)).

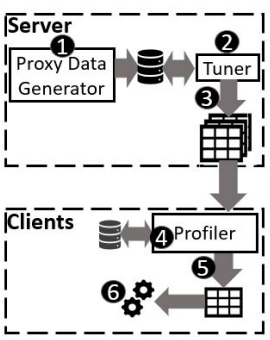
4 *FedTune*: Heterogeneity-aware Hyperparameter Optimization

With the understanding and insights grained from the above study, we propose a first of its kind privacy preserving and heterogeneity-aware hyperparameter optimization methodology, named *FedTune*.

4.1 PROXY DATASET-BASED HYPERPARAMETER CUSTOMIZATION

Considering the privacy restrictions and client side cost, offloading tuning tasks to the clients is infeasible. Therefore, we propose a proxy-based hyperparameter customization method that allows the tuning tasks to be executed on the server *without any client information nor accumulated client information.* The key rationale for our approach is similar to the spirit of transfer learning. That is given the data heterogeneity is the main property for distinguishing clients in terms of optimized hyperparameters, as long as we can create a set of proxy datasets with different heterogeneity characteristics, we can tune hyperparameters on the set of proxy datasets and then "transfer" such learned knowledge of hyperparameters on data heterogeneity to the actual clients. Specifically, we perform controllable biased sampling on a synthetic dataset (such as Cifar10 and FashionMNIST) to generate proxy datasets with different data heterogeneity characteristics. Each proxy dataset has two key characterization measures: data quantity and data distribution heterogeneity (*HI*), as explained in the previous section.

Note that the synthetic dataset used for sampling does not capture any information of clients and can be even IID data. This synthetic dataset only needs to reflect some general information such as the number of classes, key features, etc. of the application, which is usually needed anyway when building the model. In practice, such dataset can be provided by the model developers, or from user-shared/publicly available data (Guha et al. (2019); Yang et al. (2018)). For example, the initial training dataset used by the model developers when designing the architecture of the model (McMahan et al. (2021); Zhang et al. (2020). In fact, the current practice global based approach also needs a similar dataset for tuning the global hyperparameters Bonawitz et al. (2019); Yang et al. (2018); Guha et al. (2019). Thus we do not impose any additional assumptions on such dataset and no client information is needed for this dataset.



1	Proxy Data Generator	Generate proxy dataset with different quantity and distribution heterogeneity.
2	Tuner	Hyperparameter tuning on generated proxy dataset using Bayesian Strengthened Tuner.
3	Populate HRT	Populate Hyperparameter Reference Table (HRT) and send to each client.
4	Client-side Profiler	Profile data quantity and distribution heterogeneity of the client.
5	Look up HRT	Use profiled information to look up the HRT and apply the corresponding hyperparameters for training.
6	Local Training Module	Perform local training using selected hyperparameters.

Figure 3: *FedTune* system design. Shows the major steps involved in the tuning process.

4.2 BAYESIAN STRENGTHENED TUNER AND HYPERPARAMETER REFERENCE TABLE

To get the optimal hyperparameter sets for specific types of data heterogeneity (i.e. a certain combination data quantity and *HI*), we need to search our total hyperparameter space efficiently. We do this via Bayesian Optimization. For a certain number of data quantity and *HI* combinations, we trial and then record their discovered hyperparameters in a *Hyperparamter Reference Table* (HRT). The HRT is only sent *one way from server to clients*. On the client-side, we have a profiler that measures the local dataset's *HI* and data quantity, and these measures are only kept at this client and never shared with server nor other clients. Therefore, our solution *completely respects the privacy requirements* of FL and impose almost *no additional overhead* on the clients. Each client looks up the table to choose the best matching entry according to its profiler measures. The client then uses those hyperparameters for local training and the FL training proceeds as usual. A description of all the steps and a complete system overview is given in Figure 3.

Bayesian Optimization (BO) is chosen as our tuning method because it has been proven to be valuable in hyperparameter optimizations (Snoek et al. (2015); Wu et al. (2020a)). Here we leverage BO for tuning the hyperparameters in the tailored search space. BO judiciously selects the next points to explore based on the predefined acquisition function values obtained from previous exploration steps. We use EI (Expected Improvement) (Vazquez & Bect (2010)) as our acquisition function as it does not require hyperparameter tuning and it is easy for setting intuitive stop conditions. EI aims at maximizing the expected improvement from the new explorations over the current best results

and is defined as: $EI(Hp) = (y_o - \mu(Hp))\Phi(\gamma(Hp)) + \sigma(Hp)\phi(\gamma(Hp))$, where $\mu(\cdot)$ and $\sigma(\cdot)$ are predictive mean function and predictive standard deviation function, respectively; y_o is the best current value at $argmin_{(Hp)}y(Hp)$; $\gamma(Hp) = \frac{y_o - \mu(Hp)}{Hp}$; $\Phi(\cdot)$ and $\phi(\cdot)$ are predictive cumulative distribution function of standard normal and probability density function of standard normal.

Hyperparameter Reference Table is a two-dimensional array where the rows are the data distribution heterogeneity (i.e., HI) and the columns are the data quantity. Each cell contains the hyperparameter sets for datasets with the corresponding combination of HI and data quantities. The table is typically a few KBs and sent with the global model, thus the networking overhead is negligible.

Search Space is the same for each cell. Here we set the learning rate range between $0.002, 0.8$ with 0.002 increments, leading to 400 total learning rates. Batch sizes are $4, 8, 16$ and local epochs are $5, 10, 15$, so the total number of possible combinations of hyperparameters are 3,600 per cell.

Objective Function is a small FL simulator we run for 20 rounds with 5 clients. The dataset heterogeneity properties of these clients are set to those of the cell which the optimizer is running on. The FL simulator returns the final accuracy as the function output, and the BO maximizes this output.

The Tuning Process. The BO-based tuner then traverses through each of the possible combinations of HI and data quantity, searching the full hyperparameter space to find the set that gives the highest accuracy. The search for each cell stops when there was no increase in the FL simulator's accuracy in the last 5 rounds. This traversal continues until we find a hyperparameter set for each cell. Therefore, the total search space is 3,600 times the number of cells. The decision to choose the number of cells is important since it is one of the most important factors determining the total search space and thus the efficiency of the tuner. We conduct a sensitivity analysis in Appendix Figure 8.

5 Evaluation

5.1 Experiment Setup

Federated Learning Setup. We use four popular FL datasets for evaluation: FEMNIST (Caldas et al. (2018)), Cifar10/Cifar100 (Krizhevsky et al. (2009)), and Fashion-MNIST (Xiao et al. (2017)) (F-MNIST). Details of the model, training, and client setup are given in Appendix Table 3. The test results presented in this paper are derived from inferring the global model on their respective test datasets. Further details of the synthetic datasets and how their proxy dataset are sampled are given in the Appendix (G.2).

Control and Quantify Heterogeneity. Due to the lack of production level user datasets, almost all literature in FL (McMahan et al. (2021); Zawad et al. (2021); Sattler et al. (2019); Zhao et al. (2018); Briggs et al. (2020)) use controlled data distribution heterogeneity. We follow these works for our evaluation as well. The total dataset is split into smaller separate datasets which contains a specific data distribution and quantity heterogeneity (such as *HI* of 0.8 and 800 datapoints) and then assigned to a client (details are provided in Appendix G.2). This is similar to the distribution strategies used in Zawad et al. (2021); Chai et al. (2020); Li et al. (2019). Such controlled setups are usually for the purpose of a systematic characterization and clear analysis. It is worth noting that our approach does not assume any specific patterns in data distribution heterogeneity and thus can be applied to any dataset. We also conduct experiments using the Gaussian and Poisson distributions as used by other papers (Sattler et al. (2019); Briggs et al. (2020)), as well as the default distribution used in LEAF Caldas et al. (2018) to demonstrate that our approach is general and does not depend on specific heterogeneity distribution. The evaluation results of different distributions are present in the Appendix Figures 6 and 7.

Hyperparameter Optimization Methods. We compare our BO-based solution to Random Search and Grid Search baselines since there are no dedicated hyperparameter tuning frameworks for FL. For Random Search, we perform uniform random sampling from the full search space without replacement and keep the hyperparameter set that gives us the highest accuracy per cell. In Grid Search, we traverse the full space in order and only keep the best hyperparameter set. The total search space is 86,400 possible combinations of hyperparameters, as explained in the previous sections.

Testbed. We build a FL testbed using Tensorflow for the datasets by deploying each client on a cluster with its own exclusive Intel Xeon 2.2GHz CPU. The server (i.e. aggregator) is deployed on

a separate node with 40 CPUs. *FedTune*'s Bayesian Strengthened Tuner, Random Search and Grid Search are performed on the server. The server and the clients communicate their weights via sockets.

5.2 Performance Comparison

Dataset	Global Tuning	*FedTune*	Hand-Tuning
FEMNIST	74.14%	81.24%	81.64%
Cifar10	68.13%	72.32/%	72.66%
Cifar100	52.52%	56.21%	56.89%
F-MNIST	73.99%	79.73%	80.03%

Table 1: Accuracy over rounds comparison for different tunings for FEMNIST

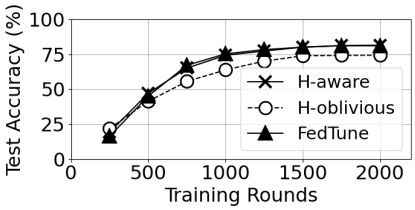
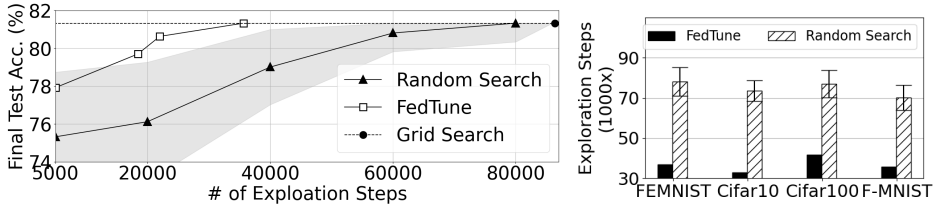


Figure 4: The test accuracy comparisons between *FedTune*, Hand-Tuned, and Global Tuning.

First, we run different datasets FEMNIST, Cifar10, Cifar100, and F-MNIST separately with the FL setting. We compare Global Tuning, *FedTune*, and Hand-tuning method, and present the best test accuracy achieved in Table 1. We can find that our proposed method *FedTune* outperforms Global Tuning by a large margin in all models. For example, on FEMNIST dataset, the test accuracy by *FedTune* improves more than 7% than Global Tuning (Figure 4). On other datasets, *FedTune* still gets better accuracy by at least 3.69% increase. On the other hand, the accuracies achieved with *FedTune* come very close to the Hand-Tuning method with less than 1% margin of error. This demonstrates that our framework can achieve a model performance on par with the best case.

5.3 Hyperparameter Optimization Cost



(a) (b)

Figure 5: (a) Test accuracy of the global model achieved with hyperparameters derived at different stages of tuning for the FEMNIST dataset. (b) Tuning iterations comparison across different datasets.

We next evaluate the efficiency of *FedTune*'s tuner compared to Random Search and Grid Search (Hand-Tuning can be considered as Grid Search). In Figure 5a, we show the test accuracies achieved by the FL system when using the hyperparameters found after searching for that particular number of steps for the FEMNIST dataset. Each step represents the number of total hyperparameter combinations searched. We observe that *FedTune*'s Bayesian Strengthened Tuner explores the space efficiently and achieves the same accuracy as the Grid Search method (81.24%) after around 36,000 steps, which is less than half the total search space. For Random Search, we show the mean accuracy vs. steps and their 95% error margin after 10 runs with different seeds. We find that while some runs initially perform better than *FedTune* at around 5,000 steps, eventually *FedTune* performs better as its improvement is steeper. The mean number of steps taken by Random Search is around 79,000 steps, which is more than twice as that for *FedTune*. The number of steps taken to achieve terminal accuracy for the other datasets for *FedTune* and Random Search with their error margins are given in Figure 5b. We see here that *FedTune* consistently outperforms Random Search across all of them.

6 Conclusion

In this paper, we demystify hyperparameter optimization in FL by identifying the opportunities and challenges via empirical experiments. Inspired by our study, we propose *FedTune*, a privacy preserving and data heterogeneity-aware hyperparameter tuning methodology for FL. The core of *FedTune* is a proxy dataset based hyperparameter customization approach that addresses the privacy and tuning cost challenges. *FedTune* outperforms baselines by up to 7% in performance with greater search efficiency.

7 REPRODUCIBILITY STATEMENT

All the training hyperparameters and experimental setups are provided in the the Experiment Setup section and Appendix. The code repo can be accessed through: https://anonymous.4open.science/r/FLTUNE-EAD4/.

REFERENCES

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

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2020.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 125–136, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Mikhail Khodak, Tian Li, Liam Li, M Balcan, Virginia Smith, and Ameet Talwalkar. Weight sharing for hyperparameter optimization in federated learning. In *Int. Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2020*, 2020.

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

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. *University of Toronto, Technical Report*, 2009.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

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.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *arXiv preprint arXiv:2006.07242*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

H Brendan McMahan et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1), 2021.

Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pp. 4615–4625. PMLR, 2019.

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

Jacquelyn K O'herrin, Norman Fost, and Kenneth A Kudsk. Health insurance portability accountability act (hipaa) regulations: effect on medical record research. *Annals of surgery*, 239(6):772, 2004.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

Jia Qian, Xenofon Fafoutis, and Lars Kai Hansen. Towards federated learning: Robustness analytics to data heterogeneity. *arXiv preprint arXiv:2002.05038*, 2020.

Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017a.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017b.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pp. 2171–2180. PMLR, 2015.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

Emmanuel Vazquez and Julien Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11): 3088–3095, 2010.

Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1698–1707. IEEE, 2020.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pp. 788–798. PMLR, 2020a.

Wentai Wu, Ligang He, Weiwei Lin, and Rui Mao. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 2020b.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data, 2021.

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

Xin Yao, Tianchi Huang, Rui-Xiao Zhang, Ruiyu Li, and Lifeng Sun. Federated learning with unbiased gradient aggregation and controllable meta updating. *arXiv preprint arXiv:1910.08234*, 2019.

Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pp. 1–5, 2015.

Syed Zawad, Ahsan Ali, Pin-Yu Chen, Ali Anwar, Yi Zhou, Nathalie Baracaldo, Yuan Tian, and Feng Yan. Curse or redemption? how data heterogeneity affects the robustness of federated learning. *arXiv preprint arXiv:2102.00655*, 2021.

Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. Engineering federated learning systems: A literature review. In *International Conference on Software Business*, pp. 210–218. Springer, 2020.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Haifeng Zheng, Min Gao, Zhizhang Chen, and Xinxin Feng. A distributed hierarchical deep computation model for federated learning in edge computing. *IEEE Transactions on Industrial Informatics*, 2021.

G SUPPLEMENTARY MATERIALS

G.1 LEARNING RATE TREND UNDER DIFFERENT DATA HETEROGENEITY

Heterogeneity Index	Batch Size / Number of Data Points			
	5/400	10/400	20/800	30/800
0.15	0.021	0.04	0.041	0.061
0.50	0.032	0.065	0.071	0.105
0.75	0.042	0.086	0.081	0.125

Table 2: Optimal learning rate under different data heterogeneity levels, batch sizes, and data sizes.

Here we demonstrate how the learning rate for the clients varies across data distribution and quantity. We observe that with a constant number of data points and batch size, increasing heterogeneity (i.e. *HI*) leads to higher optimal learning rate. Similarly, with a constant *HI*, increasing number of data points and batch size leads to increasing learning rate.

G.2 TRAINING AND PROXY DATASET SETUP

Dataset	**Model**	**Train/Test split**	**Clients Total/Per Round**	**Global LR /Batch Size**	**Training Rounds**
FEMNIST	2 conv 2 dense	49,644/6,200	192/10	0.004/8	2000
Cifar100	Resnet18	50,000/10,000	50/5	0.045/16	1000
Cifar10	4 conv 2 dense	50,000/10,000	50/5	0.05/16	500
F-MNIST	2 conv 2 dense	50,000/10,000	50/5	0.002/8	500

Table 3: Training Setup.

We perform our experiments using the popular image classification datasets Cifar100, Cifar10 and Fashion-MNIST. We also use the widely used FEMNIST dataset, which is a handwritten digit and character image classification dataset made specifically for benchmarking Federated Learning applications. It contains 62 classes and around 800,000 images split into 3,550 clients. We sample from it using the seed and sample found in their official repository in github[1]. Since it does not have a separate evaluation dataset, we use the same setup in Caldas et al. (2018) and derive a balanced test dataset of size 6,200 by randomly sampling 100 datapoints per class from the unused datapoints.

The set of proxy datasets are uniformly sampled from their full training datasets. Note that this sampled dataset is removed from the full dataset. Therefore, all proxy datasets have no overlap with either training nor testing datasets. Proxy datasets in FEMNIST, CIFAR10, CIFAR100, and FashionMNIST contain 5000, 5000, 5000, and 4000 samples, respectively. The remaining training datasets (after removing the sampled proxy datasets for training) are 44664, 45000, 45000, and 46000 samples, respectively.

We control the different data distributions within each client by splitting them into groups and subgroups. We first create 6 groups of clients by splitting them equally (e.g. in FEMNIST, 192 clients are split into 32 clients per group), and assign each of these devices to get 100/200/400/600/800/1000 data points respectively. We further split these groups into 4 more evenly split subgroups (e.g. in FEMNIST, 32 clients get split into 4 groups of 8). These groups are then assigned *HI*s of 0.2, 0.4, 0.6 and 0.8.

G.3 ROBUSTNESS TO DATA HETEROGENEITY METRICS

Apart from *HI*, state-of-the-art papers also use other methods of quantifying heterogeneity such as *Gaussian* and *Possion* distribution sampling (for both quantity and distribution heterogeneity) Reddi et al. (2020); Zawad et al. (2021). To demonstrate that *FedTune* works with other distribution metrics, we compare the accuracy increase we get after applying *FedTune* compared to Global Tuning. We do this across the heterogeneity types *HI*, *Gaussian*, and *Poisson*, and the results are presented in Figure 6.

[1] https://github.com/TalwalkarLab/leaf/

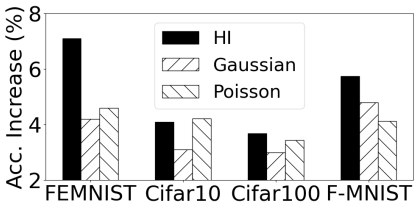


Figure 6: Final test accuracy comparison between global, *FedTune* with transferred dataset and *FedTune* on original dataset.

We observe that across all datasets, *FedTune* results in varying degrees of accuracy improvement for all different types of data heterogeneity metrics. This demonstrates that our framework is robust to different types of data distribution metrics.

G.4 Compatibility with Other Heterogeneity-aware FL Optimization

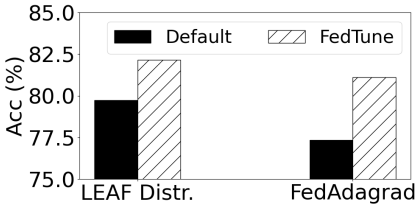


Figure 7: Final test accuracy comparison between global and *FedTune* when using LEAF's (Caldas et al. (2018)) default distribution (*LEAF Distr.*) and when used with and without *FedTune*.

We perform additional experiments to demonstrate *FedTune* is compatible with other state-of-the-art heterogeneity-aware optimizations in FL. In Figure 7, the first set of bars show the comparison of test accuracy at convergence between global tuning (*Default*) and *FedTune* when using LEAF's default distribution. We observe that using our customized hyperparameter tuning can achieve an accuracy improvement of around 2.3%. The second set of bars show the change in accuracy when using FedAdagrad (Reddi et al. (2020)) by itself (*Default*) versus adding *FedTune* on top of it (*FedTune*). We observe that with the help of *FedTune*, the final accuracy is improved around 4%, confirming that *FedTune* and FedAdagrad are complementary to each other and can be combined to achieve an even better performance.

G.5 Sensitivity Analysis of HRT Size against accuracy

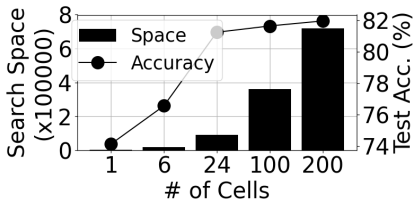


Figure 8: Sensitivity analysis of the hyperparameter search space as a function of HRT cells against test accuracy and cost using FEMNIST.

Figure 8 shows the results of the sensitivity analysis of how the number of cells in the HRT impacts the search space. For example, 1 cell means that only one set of hyperparameters is used to train the full system, i.e., a global tuning set. As we increase the number of cells, there is a drastic increase in the total search space, making it expensive to tune. Figure 8 shows how the final test accuracy for FEMNIST changes with varying number of HRT cells for our approach. It is clear

that the benefits of increasing the number of cells after 24 diminish greatly while the search space keeps on increasing. Thus, in our experiments, an HRT with 24 cell blocks strikes a good balance between search cost and accuracy. Specifically, we use HIs of $0.2, 0.4, 0.6, 0.8$ and data quantities of $100, 200, 400, 600, 800, 1000$ in our evaluation.