FedTune: Differentiable Hyperparameter Optimization for Personalized Federated Learning

Anonymous Authors[1]

Abstract

Federated Learning (FL) has gained prominence in distributed machine learning for collaborative model training while preserving data privacy. However, FL encounters challenges due to data heterogeneity, with current research primarily focused on innovative training algorithms. The potential of hyperparameter optimization (HPO) to address data heterogeneity remains underutilized. To bridge this gap, we propose FedTune, a novel differentiable HPO framework tailored to mitigate data heterogeneity issues by enabling personalized HPO for each client. By formulating personalized HPO as an optimization problem targeting joint distribution parameters within the clients' search space, FedTune leverages gradient information from differentiable validation loss to significantly enhance the efficiency of the HPO process. Experimental evaluations demonstrate that FedTune achieves optimal model performance across varying levels of data heterogeneity and accelerates the identification of optimal hyperparameters for individual clients.

1. Introduction

In recent years, Federated Learning (FL) has emerged as a significant distributed machine learning framework (Li et al., 2023). FL allows for the utilization of data from different clients without centralizing or exchanging users' raw data, promoting model enhancement and knowledge sharing while preserving privacy. However, the implementation of FL in real-world scenarios presents significant challenges due to data heterogeneity (Shen et al., 2023). The data distributions among participating clients are often non-independently and identically distributed (Non-IID), and such differences in data distributions can lead to convergence instability during training and performance degradation of the global model. While many research efforts have been devoted to addressing data heterogeneity in FL, most of them focus on developing new training architectures and algorithms (Li et al., 2020; Wang et al., 2020) or mitigating the effect of heterogeneity by preprocessing client-side local data (Wen et al., 2022). There is still a gap in research on addressing data heterogeneity from the perspective of hyperparameters of the participating clients.

Due to the variability of data distribution, different clients may have distinct optimization landscapes. It is possible that some clients' landscapes resemble steep valleys, while others' resemble gentle hills. This necessitates the customization of hyperparameter configurations for each client to enhance the optimization performance of FL and alleviate the heterogeneity among clients (Wang et al., 2022). Some existing studies have demonstrated that customizing the learning rate for each client can effectively improve the performance of the global model (Mukherjee et al., 2023). Although there are some commonly used hyperparameter optimization (HPO) methods, such as Hyperband, challenges persist in applying them to FL, including search complexity and parallel evaluation limitations.

- *Exponential Growth in Search Complexity*: The optimization of hyperparameters for each client results in exponential growth in the number of possible hyperparameter combinations as the number of clients increases, increasing the complexity of the HPO process.
- *Limitations of Parallel Evaluation*: HPO in FL is constrained by privacy concerns, and can only be evaluated on the user's personal devices. These devices typically have limited computational power and unreliable network connections, which limits the number of communication rounds and requires a sequential approach to evaluate hyperparameter configurations. In contrast, HPO in other domains can be evaluated in parallel for different configurations using computational clusters.

To address these challenges, there is an urgent need to design HPO algorithms with high search efficiency for FL.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Such algorithms can better cope with the exponential expansion of the search space and minimize the number of communication rounds required for HPO.

Differentiable search methods in Neural Architecture Search (NAS) have achieved impressive results by forming a hypernetwork of different candidate networks and introducing network architecture parameters to select between them (Baymurzina et al., 2022). By making the validation loss differentiable with respect to the network architecture parameters, differentiable NASs significantly improve the efficiency of network architecture search. Inspired by this approach, we propose **FedTune**, a novel approach that introduces the *distribution parameters* of hyperparameters. In FedTune, the hyperparameters of each client are sampled from its corresponding distribution parameters, and the evaluation mechanism of the hyperparameters is specially designed to ensure that the validation loss is differentiable with respect to the distribution parameters. We minimize interference with the existing FL workflow and compute the gradient of the distribution parameters to update them until they converge for all clients. FedTune's credit assignment mechanism assigns importance scores to the hyperparameter decisions of individual clients based on their contributions to the validation performance, which significantly improves the optimization efficiency and speeds up the convergence.

The main contributions of this paper are as follows

- We present the first original framework for differentiable personalized HPO in FL, which includes problem formulation, search space design, and hyperparameter sampling methods.
- We propose FedTune, a differentiable HPO method that enables personalized and efficient HPO while minimizing interference with the FL system and adhering to privacy protocols.
- We extensively evaluate the performance of FedTune through experiments and validate its effectiveness. Our empirical results demonstrate that FedTune achieves the fastest HPO speed and outperforms other methods in scenarios with varying degrees of data heterogeneity.

2. Related Works

2.1. Non-IID Federated Learning

In the field of FL, several solutions have been proposed to tackle the issue of data heterogeneity. Some works have focused on reducing the differences in data distribution among clients by sharing a public dataset (Huang et al., 2022) or utilizing data samples generated by an autoencoder (Wen et al., 2022). Others have proposed innovations in model training, such as the FedProx algorithm, which enhances the model's performance on Non-IID data by incorporating L2 regularization terms in the local updates (Li et al., 2020). Furthermore, at the level of learning algorithms, meta-learning (Fallah et al., 2020) and multi-task learning (Zhang et al., 2023) have been employed to improve model adaptation to heterogeneous data, e.g., using the FedAvg algorithm for basic training and fine-tuning for personalized training. Additionally, framework-level innovations, such as client-side clustering methods (Ghosh et al., 2020) and distillation-based semi-supervised FL algorithms (DS-FL) (Itahara et al., 2023), have been designed to better adapt to heterogeneous data. Although each of these approaches addresses the Non-IID problem from a different perspective, little work has been done to address data heterogeneity from the viewpoint of personalized HPO. To bridge this gap, we propose FedTune, which efficiently searches for personalized hyperparameters for each client and is compatible with other FL pipeline.

2.2. Hyperparameter Optimization

In addressing the challenges of HPO, researchers have proposed a variety of strategies, each with its specific advantages and disadvantages. Grid search methods can thoroughly explore the hyperparameter space, but they are often impractical for practical applications due to their high computational cost, which grows significantly with the number of hyperparameters (Larochelle et al., 2007). As a simpler alternative, random search tends to outperform grid search in high-dimensional spaces (Bergstra & Bengio, 2012). In contrast, bayesian optimization constructs probabilistic models to predict efficient hyperparameter combinations and provides a more intelligent search strategy that has demonstrated its sample efficiency in practice (Snoek et al., 2012). To allocate computational resources more efficiently, multi-fidelity methods such as Hyperband (Li et al., 2018) and successive halving algorithms (Jamieson & Talwalkar, 2016) are employed, allowing them to quickly eliminate less effective configurations. However, these methods are not designed with FL in mind, and they typically lack communication efficiency and scalability when the search space grows exponentially. FedEx (Khodak et al., 2021), an HPO approach designed for FL, effectively addresses the specific challenges of HPO by parallelizing hyperparametric experiments across distributed data sources, achieving a balance between privacy, communication, and computational efficiency. Nevertheless, FedEx does not support personalized HPO on a client-by-client basis but rather focuses on finding a common hyperparameter configuration that works for all participants.

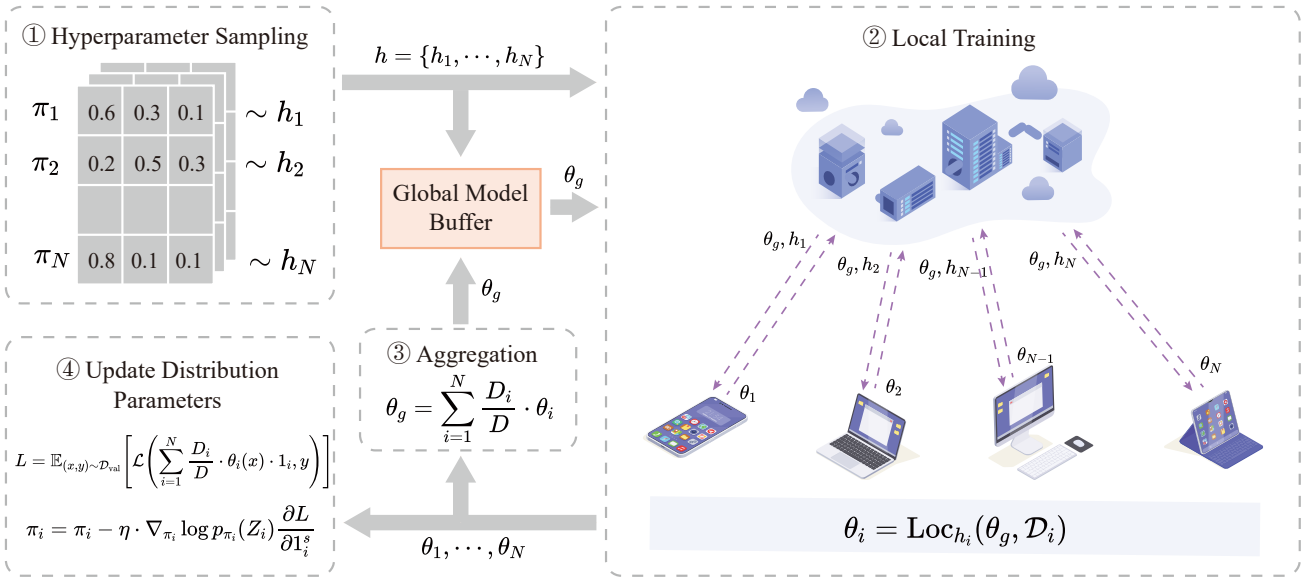

Figure 1. The framework of FedTune. In each communication round: 1. The server samples hyperparameter h_i from the corresponding distribution parameters π_i for each client. The server then obtains the global model θ_g corresponding to the current configuration $h = \{h_1, \cdots, h_N\}$ from the global model buffer $\mathcal{G}[h]$. 2. The server sends θ_g and h_i to the corresponding client c_i, and each client trains based on the hyperparameters and global model received. 3. The server aggregates the models uploaded by the clients $\theta_1, \cdots, \theta_N$ and updates the global model buffer $\mathcal{G}[h] = \theta_g$. 4. The distribution parameters of each client are updated using (11).

3. Methodology

The primary objective of FedTune is to establish an efficient and cost-effective mechanism for adaptive HPO within the FL framework while minimizing disruptions to the FL pipeline. In this section, we first present the problem formulation of personalized HPO in FL. We then delve into the search space of hyperparameters after incorporating random relaxation within a differentiable environment. Subsequently, we outline the training approach for the distribution parameters of the hyperparameters. Finally, we combine insights from reinforcement learning, specifically policy gradients, to address the credit assignment problem.

3.1. Personalized Setting and Problem Formulation

As personalization pertains to the hyperparameters associated with local training, we focus on optimizing the hyperparameters on the client side. Consider an FL system with a client set $\mathcal{C} = \{c_1, \cdots, c_N\}$ consisting of N clients, where each client possesses a training set $\mathcal{D}_i$ of size D_i, and the total size of all clients is $D = \sum_{i=1}^{N} D_i$. The server holds a validation set $\mathcal{D}_{\text{val}}$ to evaluate the performance of hyperparameter configurations. The hyperparameter configuration of the FL system is denoted as $h = \{h_1, \cdots, h_N\}$, where h_i represents the specific hyperparameters for client c_i. We assume that the hyperparameter search space is the same for each client and can be represented as $\mathcal{H} = \{h^1, \cdots, h^H\}$, consisting of H discrete hyperparameters. Therefore, the search space in the personalized setting is given by $\mathcal{H}_1 \times \cdots \times \mathcal{H}_N = \mathcal{H}^N$, which exponentially expands with the number of clients. Personalized HPO can then be formulated as solving the following bi-level optimization problem

$$\min_{h \in \mathcal{H}^N} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{val}}}[\mathcal{L}(\theta_h^*(x), y)], \tag{1}$$

$$\text{s.t.} \quad \theta_h^* = \text{Alg}_h(\{\mathcal{D}_i | i = 1, \cdots, N\}), \tag{1a}$$

where Alg_h represents the execution of the FL system for the complete training course under the hyperparameter configuration h, returning the trained global model. $\mathcal{L}$ denotes the chosen loss function for the specific task. However, executing Alg_h requires running a complete FL training course, which is computationally expensive and introduces unacceptable communication and computational overhead. Therefore, there is a demand for multi-fidelity methods to evaluate the hyperparameters while executing only a fraction of the complete training process. The most common approach is the successive halving algorithm (Jamieson & Talwalkar, 2016), which gradually eliminates poorly performing hyperparameters during the training process. Nevertheless, this hard elimination procedure generates noisy validation results, leading to the premature elimination of potentially good hyperparameters due to their less favorable early performance.

To address this issue, we transform the original bi-level op-

timization problem into the following single-level surrogate problem, enabling multi-fidelity HPO

$$\min_{h\in\mathcal{H}^N} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{val}}}\left[\mathcal{L}\left(\sum_{i=1}^{N}\frac{D_i}{D}\theta_i(x),y\right)\right], \quad \text{where } \theta_i = \text{Loc}_{h_i}(\mathcal{D}_i), \tag{2}$$

where Loc_{h_i} represents the local training phase configured by h_i, returning the updated local model parameters θ_i trained on $\mathcal{D}_i$. Since this transformation allows us to choose different hyperparameter configurations between different communication rounds, it is more flexible.

3.2. Search Space and Hyperparameter Sampling

Differentiable methods have shown remarkable effectiveness in NAS (Baymurzina et al., 2022). However, making the HPO problem differentiable is still challenging as embedding sampled hyperparameters into the forward/backward propagation graph is non-trivial. In this paper, we propose FedTune, a method that performs personalized optimization of FL hyperparameters in a differentiable manner.

The search space for personalized hyperparameters grows exponentially with the number of clients and hyperparameter types, leading to extremely high search complexity. Therefore, we improve the efficiency of HPO in FedTune by directly searching for the optimal hyperparameter distribution for each client using an approach similar to policy gradients, referred to as *search gradients* in (Xie et al., 2019).

Since the hyperparameter decision for each client is generally tractable, we represent the optimal hyperparameter distribution for each client with a distribution $p(Z_i)$. By multiplying a one-hot random variable Z_i sampled from $p(Z_i)$ with the candidate hyperparameters $\mathcal{H}$, we obtain the optimal hyperparameter decision for client c_i as

$$h_i = Z_i^T\mathcal{H}, \tag{3}$$

where $p(Z_i)$ can be fully factorized, and its factors are parameterized by π_i. The objective function (2) can be stochastically relaxed as

$$\min_{\pi\in\Pi} \mathbb{E}_{Z\sim p_\pi(Z)}\left[\mathbb{E}_{(x,y)\sim\mathcal{D}_{val}}\left[\mathcal{L}\left(\sum_{i=1}^{N}\frac{D_i}{D}\theta_i(x),y\right)\right]\right], \quad \text{where } \theta_i = \text{Loc}_{Z_i^T\mathcal{H}}(\mathcal{D}_i), \tag{4}$$

where $\pi = [\pi_1,\cdots,\pi_N]$ and $Z = [Z_1,\cdots,Z_N]$. This objective function optimizes the expected performance of hyperparameters sampled from $p(Z)$. In the next subsection, we will describe how FedTune computes the gradient of validation loss L with respect to π.

3.3. Parameter Learning for Hyperparameters

Computing the gradient of the validation loss L with respect to the parameters π is not straightforward because most hyperparameters are not directly involved in the forward/backward propagation process. This makes it challenging to embed them into the propagation graph. To address this, we transform the objective function (4) from *selecting the best hyperparameters* to *selecting models trained with different hyperparameters that perform the best*. This allows us to embed Z_i into the forward/backward propagation graph during validation by multiplying Z_i with the output of the model, instead of the hyperparameters. This transformation is intuitive because the best-performing model often implies the best hyperparameters. Thus, we can utilize Z_i, which selects the best model, to sample the hyperparameters.

Assuming no additional communication and computational costs on the client side, each client trains a separate local model for all candidate hyperparameters. The collection of local models trained by client c_i is denoted as $\Theta_i = \{\theta^1,\cdots,\theta^H\}$, and the objective function (4) can be transformed to

$$\min_{\pi\in\Pi} \mathbb{E}_{Z\sim p_\pi(Z)}\left[\mathbb{E}_{(x,y)\sim\mathcal{D}_{val}}\left[\mathcal{L}(\sum_{i=1}^{N}\frac{D_i}{D}Z_i^T\Theta_i(x),y)\right]\right]. \tag{5}$$

This transformation avoids the direct multiplication of Z_i with the hyperparameters, making L differentiable with respect to Z_i. Since Z_i is sampled from $p_{\pi_i}(Z_i)$, we relax the discrete hyperparameter distribution using the concrete distribution through reparameterization into a continuous and differentiable form

$$\begin{aligned}\tilde{Z}_i^k &= f_{\pi_i}(G_i^k)\\ &= \frac{\exp((\log\pi_i^k + G_i^k)/\lambda)}{\sum_{l=1}^{H}\exp((\log\pi_i^l + G_i^l)/\lambda)},\end{aligned} \tag{6}$$

where G_i^k is a random variable following the gumbel distribution and can be sampled from a uniform distribution, i.e., $G_i^k = -\log(-\log(U_i^k))$, where U_i^k is a random variable following the uniform distribution. In (Maddison et al., 2017), it was proven that $p(\lim_{\lambda\to 0}\tilde{Z}_i^k = 1) = \frac{\pi_i^k}{\sum_{l=1}^{N}\pi_i^l}$, ensuring the unbiasedness of this relaxation. For the surrogate loss L for each sample, we can compute its gradient with respect to π_i^k as follows

$$\frac{\partial L}{\partial \pi_i^k} = \frac{\partial L}{\partial \tilde{Z}_i}\frac{\partial \tilde{Z}_i}{\partial \pi_i^k} = \frac{\partial L}{\partial \tilde{Z}_i}(\delta(k'-k) - \tilde{Z}_i^k)\tilde{Z}_i\frac{1}{\lambda\pi_i^k}. \tag{7}$$

However, training a model for each client with all candidate hyperparameters incurs unacceptable computational and communication overhead. Inspired by DSNAS (Hu

et al., 2020), we identify that this issue mainly arises from $\tilde{Z}_i$ being a soft one-hot vector. If we make $\tilde{Z}_i$ a strict one-hot vector, we can avoid training for the hyperparameters corresponding to 0 in $\tilde{Z}_i$, significantly reducing the computational and communication costs. We will elaborate on this in the next subsection.

3.4. Credit Assignment

Credit assignment is a crucial aspect of reinforcement learning as it significantly impacts policy learning performance. However, as discussed in (Xie et al., 2019), personalized HPO can also be viewed as tasks with delayed rewards, as decisions for all clients must be made before receiving a reward. This hinders convergence, which is unacceptable in FL scenarios with limited computational and communication resources. In this subsection, we discuss the credit allocated by FedTune to each client's hyperparameter decisions.

As discussed in the previous subsection, the decision variable $\tilde{Z}_i$ is differentiable with respect to the distribution parameter π_i. Therefore, we can derive the expected gradient of the decision variable with respect to the distribution parameter as follows

$$\mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}\left[\frac{\partial \tilde{Z}_i}{\partial \pi_i}\right] = \mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}[\nabla_{\pi_i} \log p_{\pi_i}(\tilde{Z}_i)\tilde{Z}_i] \tag{8}$$

From this, we can obtain the search gradient of distribution parameter π_i as

$$\begin{aligned}\mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}\left[\frac{\partial L}{\partial \pi_i}\right] &= \mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}\left[\frac{\partial L}{\partial \tilde{Z}_i}\frac{\partial \tilde{Z}_i}{\partial \pi_i}\right] \\ &= \mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}\left[\nabla_{\pi_i} \log p_{\pi_i}(\tilde{Z}_i)\frac{\partial L}{\partial \tilde{Z}_i}\tilde{Z}_i\right].\end{aligned} \tag{9}$$

As previously discussed, the use of $\tilde{Z}_i$ as a soft one-hot vector introduces computationally and communication-costly challenges to HPO. To make FedTune more feasible, we set $\lambda \to 0$ to transform $\tilde{Z}_i$ into a strict one-hot vector Z_i. This yields

$$\begin{aligned}&\lim_{\lambda \to 0}\left\{\mathbb{E}_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)}\left[\nabla_{\pi_i} \log p_{\pi_i}(\tilde{Z}_i)\frac{\partial L}{\partial \tilde{Z}_i}\tilde{Z}_i\right]\right\} \\ &= \mathbb{E}_{Z_i \sim p_{\pi_i}(Z_i)}\left[\nabla_{\pi_i} \log p_{\pi_i}(Z_i)\frac{\partial L}{\partial 1_i^s}\right],\end{aligned} \tag{10}$$

where Z_i represents a strict one-hot vector, and 1_i^s indicates the entry in Z_i whose value is 1. In this case, only the hyperparameter corresponding to the selected entry in Z_i needs to be differentiated, while the hyperparameters with 0 entries in Z_i no longer require training, significantly reducing computational and communication costs. The policy π_i can be updated using Monte Carlo methods, and the iterative formula is defined as follows

$$\pi_i = \pi_i - \eta \cdot \nabla_{\pi_i} \log p_{\pi_i}(Z_i)\frac{\partial L}{\partial 1_i^s}. \tag{11}$$

where η denotes the learning rate of π_i. Clearly, the search gradient is equivalent to the policy gradient of the distribution parameter π_i, and its credit is assigned as

$$R_i = -\frac{\partial \mathcal{L}}{\partial 1_i^s}. \tag{12}$$

This reward allocates importance scores to different hyperparameter decisions, allowing decisions that contribute more to the validation performance to receive higher rewards. This importance allocation strategy also explains why FedTune is more efficient.

Algorithm 1 Training Process of FedTune

Initialize the global model buffer $\mathcal{G}$.
for round t in $\{1, \cdots, T\}$ **do**
  Sample Z_i for each client: $Z_i \sim p_{\pi_i}(Z_i)$.
  Obtain local hyperparameters for each client: $h_i = Z_i^T \mathcal{H}$.
  Retrieve the current hyperparameter configuration: $h = \{h_1, \cdots, h_N\}$.
  Retrieve the global model for the current configuration: $\theta_g = \mathcal{G}[h]$.
  Broadcast θ_g and h_i to the corresponding clients.
  for each client in parallel **do**
    Perform local training: $\theta_i = Loc_{h_i}(\theta_g, \mathcal{D}_i)$.
    Upload θ_i to the server.
  end for
  Aggregate local models: $\theta_g = \sum_{i=1}^{N} \frac{D_i}{D}\theta_i$.
  Update the global model buffer: $\mathcal{G}[h] = \theta_g$.
  for each (x, y) sampled from $\mathcal{D}_{\text{val}}$ **do**
    Compute the validation loss: $L = \mathcal{L}\left(\sum_{i=1}^{N} \frac{D_i}{D} \cdot \theta_i(x) \cdot 1_i, y\right)$.
    Calculate the gradient of the dummy one: $\frac{\partial L}{\partial 1_i}$.
  end for
  Update distribution parameters using Equation (11).
end for

4. Evaluations

4.1. Experimental Setup

4.1.1. Baseline Algorithms

We evaluated the performance of FedTune by comparing it with commonly used HPO algorithms, including Random Search, Hyperband, and Bayesian Optimization.

- **Random Search:** The server randomly selects hyperparameters for each client, a total of 30 groups, and evaluates each group of hyperparameters with different fidelity according to the communication budget. Specifically, when the communication budget is 30 rounds, then each group of hyperparameters is evaluated with only 1 round of training, while when the communication budget is 300 rounds, then each group of hyperparameters is evaluated with 10 rounds of training.
- **Hyperband:** The server randomly selects hyperparameters for each client and schedules each trial using the hyperband algorithm. Specifically, hyperband aborts poorly performing trials early to conserve communication and computational resources. When hyperband has aborted the current trial, the server determines whether the communication budget is exhausted, and if not, randomly selects the next set of hyperparameters for training until the communication budget is exhausted.
- **Bayesian Optimization:** The server adopts Bayesian optimization algorithm to select hyperparameters for each client, and selects 30 groups in total, and evaluates each group of hyperparameters with different fidelity according to the communication budget. Specifically, when the communication budget is 30 rounds, then each group of hyperparameters is evaluated with only 1 round of training, while when the communication budget is 300 rounds, then each group of hyperparameters is evaluated with 10 rounds of training.

At the end of the HPO phase, the best performing group of hyperparameters will be identified and used for 400 rounds of training to assess the quality of the selected hyperparameters.

4.1.2. Models and Datasets

We conducted experiments using general models and datasets. To better simulate the real-world FL scenario, where the dataset exhibits Non-IID distribution among clients, we split the dataset following the dirichlet distribution with parameter α. A smaller $alpha$ value indicates a more pronounced Non-IID characteristic, and vice versa.

- **CIFAR10:** The CIFAR10 dataset consists of 60,000 color images divided into 10 categories, each with a resolution of 32x32 pixels and three color channels. The dataset is divided into 50,000 training images and 10,000 test images (Krizhevsky et al., 2009).
- **FMNIST:** The FMNIST dataset contains 70,000 grayscale images of fashion items belonging to 10 different categories. Each image has a resolution of 28x28 pixels and is preprocessed to ensure consistent scaling and orientation. The dataset is evenly distributed with 7,000 images per category (Xiao et al., 2017).
- **MNIST:** The MNIST dataset contains 70,000 grayscale images of handwritten numbers. Each image has a resolution of 28x28 pixels and is divided into 10 categories. The dataset is further divided into 60,000 training images and 10,000 test images (Lecun et al., 1998).

In our experiments, we use the commonly used 2-layer Multi-Layer Perceptron (MLP) containing a hidden layer with 128 neurons and LeNet-5 (Lecun et al., 1998) as the model for evaluation.

4.1.3. Settings

All methods were implemented using PyTorch and executed on a 64-bit Ubuntu 20.04 operating system equipped with 8 A40 GPUs. The FL pipeline used in the experiments is FedAvg. Candidate hyperparameters are learning rates [1e-1, 5e-2, 1e-2, 5e-3, 1e-3] and weight decay values [0, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5]. To ensure a fair comparison, all methods conduct hyperparameter optimization within the same communication budget. After exhausting the communication budget, 400 rounds of full-fidelity training using the best performing hyperparameters are performed to evaluate the HPO performance of the different algorithms. The reported experimental results are averaged over 30 independent runs.

4.2. Results

We present the evaluation results from various perspectives. Due to space constraints, we primarily report the results for Lenet-5 on the CIFAR-10 dataset, with additional experimental results detailed in Appendix C.

4.2.1. Comparison with Baselines

We evaluated the HPO capability of FedTune under various communication budgets. Specifically, our experiment involved 10 clients participating in the training process, with the dataset's α parameter set to 0.1. We set the communication budgets to [30, 60, 90, 120, 150, 180, 210, 240, 270, 300] and performed HPO within the limited number of

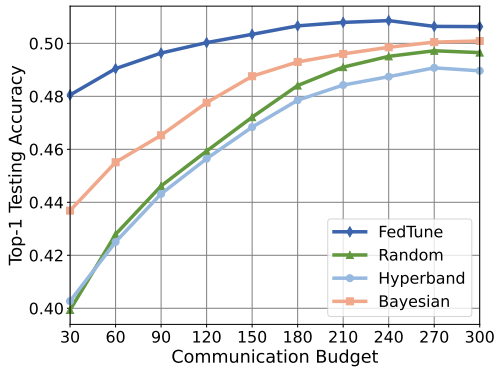

Figure 2. Top-1 test accuracy of hyperparameters searched under different communication budgets. After exhausting the communication budget, 400 rounds of training were performed based on the selected hyperparameters, followed by evaluating model performance on the test set. The system includes 10 clients, and the alpha parameter of the dataset is 0.1. It can be observed that FedTune outperforms the baseline significantly when the communication budget is limited.

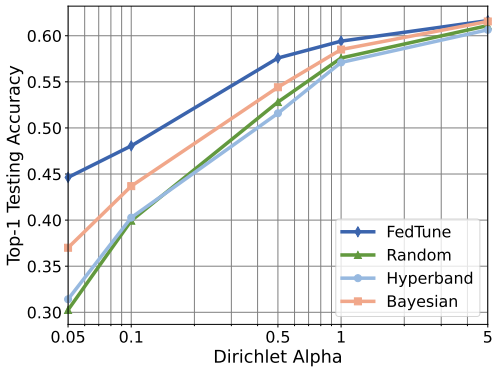

Figure 3. Top-1 test accuracy of hyperparameters searched on datasets with different alpha parameters. After exhausting the 30-round communication budget, 400 rounds of training were performed based on the selected hyperparameters, followed by evaluating model performance on the test set. The system includes 10 clients. It can be observed that FedTune outperforms the baseline significantly when the dataset is highly heterogeneous.

communication rounds specified by these budgets. As depicted in Figure 2, the optimization algorithms progressively selected better hyperparameters as the communication budget increased. However, FedTune consistently achieved the best HPO performance across all communication budgets, resulting in the training of better-performing models. Notably, when the communication budget was set to 30 rounds, FedTune outperformed Random Search, Hyperband, and Bayesian Optimization by 20.32%, 19.32%, and 9.99%, respectively. This improvement can be mainly attributed to the differentiability of FedTune, which significantly enhances optimization efficiency by computing the search gradient of the distribution parameters. Additionally, FedTune's credit assignment mechanism assigns varying importance scores to each client's hyperparameter decision, further accelerating the convergence of the distribution parameters. Consequently, FedTune achieves superior HPO results even when the communication budget is limited.

4.2.2. Effect of Data Heterogeneity

To evaluate the robustness of FedTune across different degrees of Non-IID datasets, we partitioned the dataset using Dirichlet distributions with varying α parameters and assessed the optimization performance of FedTune. Specifically, our experiments involved 10 clients participating in the training process, with a communication budget set to 30 rounds. We set the α parameters to [0.05, 0.1, 0.5, 1, 5.0] and performed HPO within the communication rounds constrained by the budget. As shown in Figure 3, the model performance progressively improves as the α increases. This trend can be attributed to the decreasing heterogeneity of the dataset, enabling the training of better-performing models. However, higher α values do not directly facilitate hyperparameter search. Importantly, FedTune consistently exhibits superior HPO performance across all tested α settings. For instance, at $\alpha = 0.05$, FedTune outperforms Random Search, Hyperband, and Bayesian Optimization algorithms by 47.51%, 42.07%, and 20.62%, respectively. These results indicate the effectiveness of FedTune in datasets with varying degrees of Non-IID and its superior adaptability compared to other baseline algorithms, particularly in highly heterogeneous dataset environments.

4.2.3. Effect of Client Scale

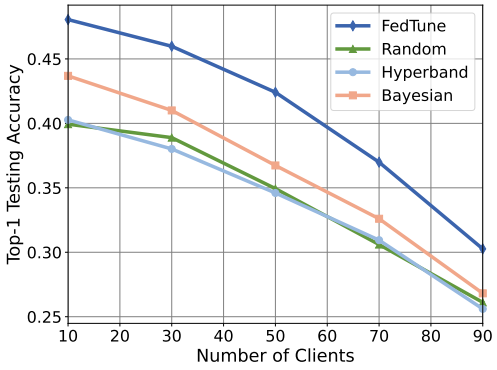

Figure 4. Top-1 test accuracy of hyperparameters searched in FL systems with different client scales. After exhausting the 30-round communication budget, 400 rounds of training were performed based on the selected hyperparameters, followed by evaluating model performance on the test set. The alpha parameter of the dataset is 0.1. It can be observed that FedTune outperforms the baseline in systems with different client scales.

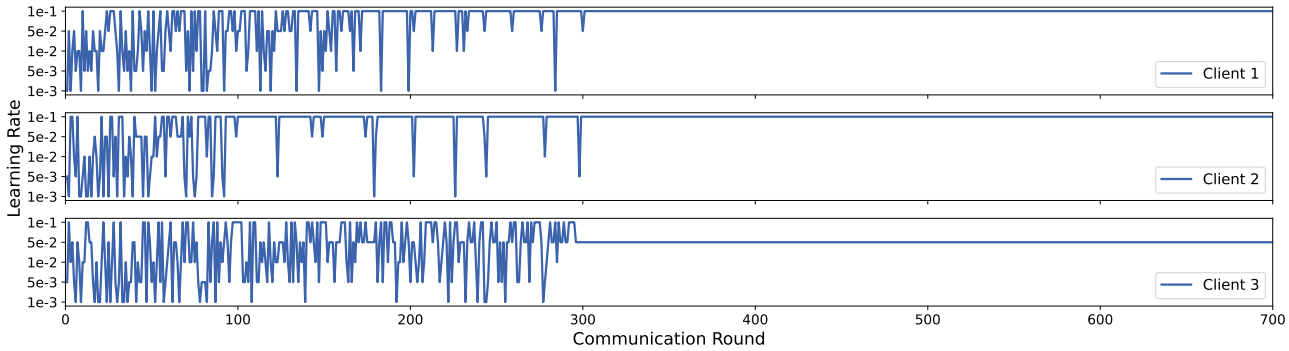

Figure 5. The convergence process of distribution parameters in FedTune. We plotted the category where the maximum value of distribution parameters occurs in each communication round. The system includes 3 clients, the alpha parameter of the dataset is 0.1, and the communication budget is 300 rounds. It can be observed that as the training progresses, the fluctuations of the category where the maximum value of distribution parameters occurs gradually decrease, indicating the convergence of FedTune.

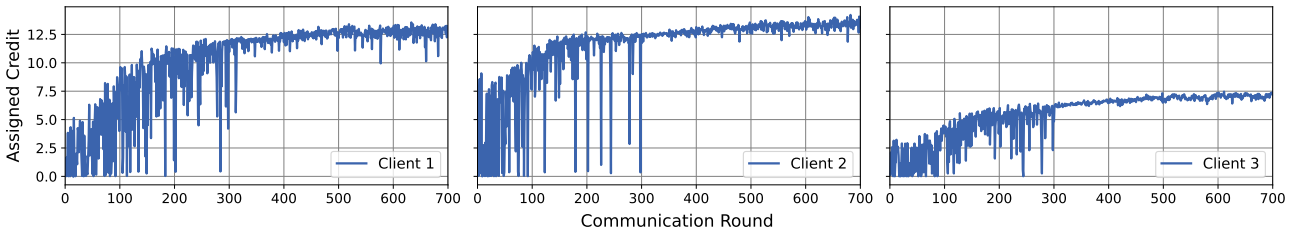

Figure 6. The convergence process of credits for each client in FedTune. The system includes 3 clients, the alpha parameter of the dataset is 0.1, and the communication budget is 300 rounds. It can be observed that as the training progresses, the credits for each client gradually increase and converge, while client 3 receives less credit, indicating that client 3 contributes less to the model validation performance.

In personalized HPO for FL, the hyperparameter search space expands exponentially with the number of clients. Therefore, we compare the optimization performance of FedTune with other baseline methods across different client scales. Specifically, in our experiment, the communication budget is set to 30, and the dataset's α parameter is 0.1. We vary the number of clients participating in the training process, considering values of [10, 30, 50, 70, 90], and perform HPO within the limited number of communication rounds specified by the budget. As illustrated in Figure 4, the model's performance progressively deteriorates as the number of clients increases. This degradation can be attributed to two main factors: first, the rapid expansion of the hyperparameter search space, which increases the search complexity, and second, the increased data dispersion as the number of clients grows, leading to fewer data samples available for each client and consequently impacting the model's performance. Nevertheless, FedTune consistently exhibits superior HPO performance across different client scales. Notably, when 90 clients are included in the FL system, FedTune achieves performance improvements of 15.90%, 18.19%, and 12.87% over Random Search, Hyperband, and Bayesian Optimization, respectively. These results indicate that FedTune demonstrates greater adaptability and stronger optimization capabilities compared to other baseline algorithms.

4.2.4. Convergence of FedTune

This subsection evaluates the convergence of FedTune. Specifically, we consider FL systems consisting of three clients, and FedTune determines the optimal learning rate for each client from a predefined set [1e-1, 5e-2, 1e-2, 5e-3, 1e-3]. In the experiment, the communication budget is set to 300, and the dataset's α parameter is 0.1. Figure 5 presents the category where the maximum value of the distribution parameter of each client is located in each communication round. It is observed that the distribution parameters of client 1 and client 2 gradually converge to select learning rates of 1e-1, while client 3 converges to select learning rates of 5e-2. During the initial phase with fewer than 100 communication rounds, the hyperparameters selected by each client exhibit frequent changes, indicating that FedTune has not yet converged and active exploration is underway to de-

termine the optimal hyperparameters. However, after more than 200 rounds of communication, the variation in hyperparameters significantly decreases, suggesting that FedTune is approaching convergence, has a clearer assessment of the optimal hyperparameters for each client, and is therefore more inclined towards exploitation rather than exploration. Figure 6 illustrates the credits assigned by FedTune to each client in each communication round. It can be observed that the amount of credits received by each client increases with the number of communication rounds, eventually reaching convergence after 300 rounds. Notably, client 3 receives fewer credits compared to the other clients, indicating its relatively lower contribution to the global model's validation performance.

5. Conclusion

This paper presents FedTune, a novel differentiable HPO framework specifically designed for FL. FedTune models the HPO process as the optimization of distribution parameters across clients and incorporates credit assignment to significantly improve optimization efficiency. Extensive experiments validate the superior performance of FedTune over the baseline approach in environments with strong data heterogeneity. Furthermore, FedTune achieves substantial performance gains even with limited budgets, data heterogeneity, and increasing client scales. However, an existing limitation of FedTune is its reliance on the assumption that the server has a validation set, which restricts its application scenarios. We leave it as future work to explore methods for eliminating this assumption.

References

Baymurzina, D., Golikov, E., and Burtsev, M. A review of neural architecture search. *Neurocomputing*, 474:82–93, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.12.014.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3557–3568. Curran Associates, Inc., 2020.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33: 19586–19597, 2020.

Hu, S., Xie, S., Zheng, H., Liu, C., Shi, J., Liu, X., and Lin, D. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12084–12092, 2020.

Huang, W., Ye, M., and Du, B. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10143–10153, June 2022.

Itahara, S., Nishio, T., Koda, Y., Morikura, M., and Yamamoto, K. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2023. doi: 10.1109/TMC.2021.3070013.

Jamieson, K. and Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 240–248, Cadiz, Spain, 09–11 May 2016. PMLR.

Khodak, M., Tu, R., Li, T., Li, L., Balcan, M.-F. F., Smith, V., and Talwalkar, A. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *Advances in Neural Information Processing Systems*, 34: 19184–19197, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 473–480, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273556.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., and He, B. A survey on federated learning systems: Vision, hype and reality for data privacy and protection.

IEEE Transactions on Knowledge and Data Engineering, 35(4):3347–3366, 2023. doi: 10.1109/TKDE.2021.3124599.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In Dhillon, I., Papailiopoulos, D., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Mukherjee, S., Loizou, N., and Stich, S. U. Locally adaptive federated learning via stochastic polyak stepsizes. *arXiv preprint arXiv:2307.06306*, 2023.

Shen, J., Cheng, N., Wang, X., Lyu, F., Xu, W., Liu, Z., Aldubaikhy, K., and Shen, X. Ringsfl: An adaptive split federated learning towards taming client heterogeneity. *IEEE Transactions on Mobile Computing*, pp. 1–16, 2023. doi: 10.1109/TMC.2023.3309633.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

Wang, Z., Kuang, W., Zhang, C., Ding, B., and Li, Y. Fedhpo-b: A benchmark suite for federated hyperparameter optimization. *arXiv preprint arXiv:2206.03966*, 2022.

Wen, H., Wu, Y., Li, J., and Duan, H. Communication-efficient federated data augmentation on non-iid data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3377–3386, June 2022.

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

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

Zhang, H., Tao, M., Shi, Y., Bi, X., and Letaief, K. B. Federated multi-task learning with non-stationary and heterogeneous data in wireless networks. *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023. doi: 10.1109/TWC.2023.3301611.

A. Derivative 1

B. Derivative 2

$$
\begin{aligned}
& E_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)} \left[\frac{\partial \tilde{Z}_i}{\partial \pi_i} \right] \\
&= E_{U_i \sim Uniform} \left[\frac{\partial f_{\pi_i}(-\log(-\log(U_i)))}{\partial \pi_i} \right] \\
&= \int_0^1 p(U_i) \frac{\partial f_{\pi_i}(-\log(-\log(U_i)))}{\partial \pi_i} \mathrm{d}U_i \\
&= \frac{\partial}{\partial \pi_i} \int p_{\pi_i}(\tilde{Z}_i) \tilde{Z}_i \mathrm{d}\tilde{Z}_i \\
&= \int p_{\pi_i}(\tilde{Z}_i) \frac{\partial \log p_{\pi_i}(\tilde{Z}_i)}{\partial \pi_i} \tilde{Z}_i \mathrm{d}\tilde{Z}_i \\
&= E_{\tilde{Z}_i \sim p_{\pi_i}(\tilde{Z}_i)} [\nabla_{\pi_i} \log p_{\pi_i}(\tilde{Z}_i) \tilde{Z}_i]
\end{aligned}
\tag{13}
$$

C. More Experiment Results

C.1. Comparison with Baselines

Table 1. Top-1 Test Accuracy on Different Datasets.

Model	Dataset	Algorithm	Communication Budget 30	60	90	120	150	180	210	240	270	300
Lenet-5	CIFAR10	FedTune	**48.05 ± 2.28**	**49.05 ± 1.74**	**49.63 ± 3.40**	**50.03 ± 5.00**	**50.34 ± 1.72**	**50.66 ± 3.20**	**50.79 ± 2.01**	**50.86 ± 0.79**	**50.64 ± 3.14**	**50.63 ± 3.89**
		Random	39.94 ± 4.42	42.79 ± 3.83	44.62 ± 4.91	45.93 ± 2.19	47.21 ± 3.23	48.41 ± 1.63	49.1 ± 0.98	49.51 ± 1.79	49.72 ± 1.32	49.65 ± 1.18
		Hyperband	40.27 ± 4.17	42.50 ± 6.22	44.32 ± 1.48	45.65 ± 2.86	46.84 ± 1.80	47.85 ± 2.78	48.43 ± 1.52	48.74 ± 0.55	49.07 ± 1.38	48.96 ± 0.85
		Bayesian	43.69 ± 3.93	45.51 ± 4.60	46.53 ± 5.81	47.76 ± 2.04	48.76 ± 1.88	49.30 ± 1.60	49.60 ± 1.37	49.85 ± 1.75	50.05 ± 0.77	50.09 ± 1.19
	FMNIST	FedTune	**81.99 ± 3.69**	**82.80 ± 3.15**	**83.42 ± 1.91**	**83.92 ± 4.49**	**84.14 ± 2.81**	**84.25 ± 0.78**	**84.73 ± 2.61**	**84.14 ± 0.68**	**85.13 ± 1.50**	84.41 ± 2.07
		Random	66.92 ± 7.92	72.23 ± 7.10	75.92 ± 4.43	78.74 ± 1.81	80.30 ± 2.54	81.48 ± 1.99	81.69 ± 2.28	81.96 ± 3.43	82.57 ± 2.26	82.65 ± 1.28
		Hyperband	65.53 ± 5.76	72.36 ± 6.33	75.45 ± 3.38	78.02 ± 1.75	79.07 ± 3.79	80.19 ± 2.10	80.70 ± 4.27	81.44 ± 2.50	81.54 ± 2.70	81.96 ± 1.03
		Bayesian	60.56 ± 6.35	67.90 ± 7.31	74.41 ± 3.31	78.28 ± 3.81	80.90 ± 3.98	82.56 ± 2.72	83.01 ± 2.47	83.77 ± 2.44	84.08 ± 1.44	**84.68 ± 1.57**
	MNIST	FedTune	**96.00 ± 4.51**	**96.66 ± 0.64**	**96.86 ± 3.07**	**97.13 ± 4.21**	**97.45 ± 0.57**	**97.65 ± 1.82**	**98.10 ± 0.29**	**97.98 ± 2.32**	**97.74 ± 0.34**	**98.02 ± 0.53**
		Random	84.92 ± 5.66	89.94 ± 4.77	93.50 ± 5.57	95.04 ± 1.89	95.86 ± 3.10	96.54 ± 0.56	97.37 ± 0.65	97.22 ± 0.79	97.48 ± 1.25	97.81±0.53
		Hyperband	84.26 ± 5.60	90.78 ± 2.26	93.75 ± 2.79	94.59 ± 2.17	95.57 ± 1.19	96.03 ± 1.64	96.49 ± 0.67	96.90 ± 0.70	97.04 ± 0.43	97.04 ± 0.59
		Bayesian	88.47 ± 6.68	92.17 ± 4.26	94.75 ± 3.10	95.79 ± 3.60	96.52 ± 4.12	97.25 ± 1.39	97.59 ± 1.21	97.90 ± 1.68	97.43 ± 0.47	97.86 ± 0.52
MLP	CIFAR10	FedTune	**40.57 ± 2.57**	**41.40 ± 2.58**	**41.90 ± 1.32**	**41.64 ± 1.55**	**42.15 ± 1.57**	**42.21 ± 1.38**	**42.45 ± 1.99**	**42.18 ± 2.36**	**42.58 ± 1.91**	**42.56 ± 0.94**
		Random	36.02 ± 2.59	37.92 ± 1.78	39.32 ± 1.73	40.42 ± 1.42	41.09 ± 1.46	41.25 ± 1.36	41.5 ± 0.88	41.16 ± 1.47	41.49 ± 0.84	41.35 ± 0.73
		Hyperband	36.56 ± 2.36	38.06 ± 1.81	39.07 ± 1.30	39.89 ± 1.14	40.74 ± 1.65	40.68 ± 0.61	40.84 ± 1.49	40.77 ± 0.70	41.02 ± 0.96	40.42 ± 0.92
		Bayesian	38.78 ± 2.43	39.03 ± 2.25	40.21 ± 1.70	41.07 ± 1.01	41.50 ± 1.18	41.64 ± 1.15	41.92 ± 0.84	41.76 ± 0.63	41.55 ± 0.88	41.93 ± 0.82
	FMNIST	FedTune	**78.50 ± 3.49**	**79.75 ± 5.18**	**80.65 ± 5.03**	**80.97 ± 2.29**	**81.52 ± 2.08**	**81.68 ± 2.86**	**81.45 ± 3.04**	**81.90 ± 4.47**	**82.49 ± 2.22**	**82.03 ± 2.08**
		Random	74.89 ± 3.78	77.36 ± 3.62	78.67 ± 2.27	79.35 ± 2.40	80.12 ± 1.25	80.94 ± 1.99	81.26 ± 1.41	81.31 ± 1.31	80.70 ± 1.21	81.29 ± 1.43
		Hyperband	74.79 ± 5.48	77.38 ± 3.19	78.47 ± 2.40	79.46 ± 1.67	79.73 ± 1.68	79.83 ± 1.15	80.27 ± 1.09	80.66 ± 1.26	80.64 ± 1.60	80.71 ± 1.26
		Bayesian	73.01 ± 2.65	75.67 ± 3.70	77.75 ± 2.28	78.85 ± 2.12	79.98 ± 1.52	80.71 ± 1.28	80.75 ± 2.03	81.19 ± 1.39	81.72 ± 0.86	81.83 ± 0.88
	MNIST	FedTune	**89.28 ± 3.33**	**90.37 ± 2.86**	**90.99 ± 1.62**	**91.24 ± 1.74**	**91.64 ± 2.59**	**91.98 ± 3.82**	**92.05 ± 3.44**	**92.36 ± 1.54**	**92.52 ± 1.46**	**92.81 ± 1.00**
		Random	83.29 ± 3.70	86.13 ± 4.29	87.88 ± 3.75	89.41 ± 2.88	90.20 ± 2.12	90.95 ± 1.97	91.03 ± 1.30	91.5 ± 2.55	91.96 ± 2.91	92.47 ± 3.22
		Hyperband	83.66 ± 5.87	86.38 ± 4.61	87.33 ± 4.15	88.93 ± 2.64	90.03 ± 1.85	90.35 ± 1.66	90.73 ± 2.53	90.74 ± 2.07	91.44 ± 0.90	91.71 ± 1.53
		Bayesian	85.71 ± 4.28	87.60 ± 3.61	88.91 ± 5.30	89.96 ± 2.06	90.88 ± 3.42	91.31 ± 2.87	91.59 ± 2.32	91.82 ± 2.56	92.37 ± 3.33	92.08 ± 1.54

C.2. Effect of Dataset Distribution

C.3. Effect of Client Scale

Table 2. Top-1 Accuracy on Different Datasets. Model Lenet-5.

Dataset	Algorithm	Dirichlet Alpha $\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 5$
CIFAR10	FedTune	**44.63 ± 3.55**	**48.05 ± 2.28**	**57.58 ± 1.09**	**59.41 ± 0.61**	**61.64 ± 0.42**
	Random	30.26 ± 5.99	39.94 ± 4.42	52.81 ± 2.86	57.59 ± 1.55	61.10 ± 0.62
	Hyperband	31.41 ± 5.90	40.27 ± 4.17	51.59 ± 2.98	57.10 ± 1.76	60.64 ± 1.33
	Bayesian	37.00 ± 4.57	43.69 ± 3.93	54.41 ± 3.99	58.51 ± 2.40	61.55 ± 1.30
FMNIST	FedTune	**71.31 ± 6.47**	**81.99 ± 3.69**	**89.06 ± 0.61**	**90.29 ± 0.10**	**90.82 ± 0.10**
	Random	57.42 ± 6.70	66.92 ± 7.92	80.05 ± 3.21	86.54 ± 1.14	89.26 ± 0.38
	Hyperband	53.82 ± 6.28	65.53 ± 5.76	83.57 ± 1.83	85.07 ± 1.52	88.55 ± 0.64
	Bayesian	52.00 ± 7.18	60.56 ± 6.35	79.73 ± 5.35	85.21 ± 2.18	86.03 ± 1.78
MNIST	FedTune	**89.36 ± 7.52**	**96.00 ± 4.51**	**98.94 ± 0.09**	**99.12 ± 0.04**	**99.23 ± 0.02**
	Random	61.30 ± 8.39	84.92 ± 5.66	95.91 ± 1.61	97.76 ± 0.42	98.57 ± 0.19
	Hyperband	57.54 ± 10.48	84.26 ± 5.60	95.07 ± 2.19	97.34 ± 1.15	98.58 ± 0.17
	Bayesian	69.51 ± 9.27	88.47 ± 6.68	95.14 ± 1.66	97.31 ± 0.67	98.39 ± 0.27

Table 3. Top-1 Accuracy on Different Datasets. Model MLP.

Dataset	Algorithm	Dirichlet Alpha $\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 5$
CIFAR10	FedTune	**36.64 ± 2.71**	**40.57 ± 2.57**	46.89 ± 2.03	**50.62 ± 0.61**	**52.52 ± 0.85**
	Random	31.94 ± 3.04	36.02 ± 2.59	46.54 ± 0.90	49.89 ± 0.77	51.43 ± 0.37
	Hyperband	33.80 ± 2.91	36.56 ± 2.36	47.06 ± 1.08	48.64 ± 0.75	51.21 ± 0.59
	Bayesian	33.34 ± 1.94	38.78 ± 2.43	**47.52 ± 0.52**	48.10 ± 1.08	50.11 ± 0.76
FMNIST	FedTune	**67.00 ± 7.42**	**78.50 ± 3.49**	**84.05 ± 1.18**	85.67 ± 0.96	85.52 ± 1.05
	Random	61.35 ± 5.84	74.89 ± 3.78	82.29 ± 2.58	85.81 ± 0.85	85.30 ± 0.89
	Hyperband	63.09 ± 5.13	74.79 ± 5.48	82.17 ± 2.29	85.48 ± 1.22	**85.79 ± 0.78**
	Bayesian	62.03 ± 6.67	73.01 ± 2.65	83.18 ± 1.30	**85.84 ± 0.73**	85.50 ± 0.91
MNIST	FedTune	**80.35 ± 5.77**	**89.28 ± 3.33**	**95.66 ± 0.71**	**96.39 ± 0.71**	**97.06 ± 0.37**
	Random	69.96 ± 5.70	83.29 ± 3.70	92.71 ± 1.74	94.02 ± 1.72	96.51 ± 0.48
	Hyperband	70.72 ± 3.05	83.66 ± 5.87	93.05 ± 1.99	94.06 ± 1.73	95.67 ± 0.64
	Bayesian	74.38 ± 7.57	85.71 ± 4.28	93.41 ± 2.53	93.45 ± 2.33	96.21 ± 0.89

Table 4. Top-1 Accuracy on Different Datasets. Model Lenet-5.

Dataset	Algorithm	Number of Clients 10	30	50	70	90
CIFAR10	FedTune	**48.05 ± 2.28**	**45.98 ± 6.13**	**42.41 ± 6.23**	**36.99 ± 5.55**	**30.26 ± 7.01**
	Random	39.94 ± 4.42	38.89 ± 5.06	34.94 ± 3.98	30.60 ± 4.95	26.11 ± 5.56
	Hyperband	40.27 ± 4.17	38.02 ± 4.05	34.60 ± 3.91	30.94 ± 3.28	25.61 ± 2.27
	Bayesian	43.69 ± 3.93	41.00 ± 5.59	36.73 ± 5.59	32.60 ± 4.23	26.81 ± 4.90
FMNIST	FedTune	**81.99 ± 3.69**	**80.53 ± 2.91**	**79.49 ± 3.07**	**76.99 ± 1.63**	**77.94 ± 2.94**
	Random	66.92 ± 7.92	65.31 ± 4.56	64.36 ± 4.86	62.18 ± 3.89	61.27 ± 4.36
	Hyperband	65.53 ± 5.76	65.16 ± 7.36	64.79 ± 3.81	61.55 ± 4.02	60.99 ± 2.15
	Bayesian	60.56 ± 6.35	60.87 ± 4.78	59.46 ± 4.81	58.06 ± 2.90	58.72 ± 5.16
MNIST	FedTune	**96.00 ± 4.51**	**97.57 ± 0.46**	**97.20 ± 0.50**	**94.86 ± 2.59**	**94.25 ± 0.82**
	Random	84.92 ± 5.66	84.50 ± 3.93	83.94 ± 1.14	83.86 ± 2.92	82.19 ± 3.67
	Hyperband	84.26 ± 5.60	84.34 ± 3.48	83.75 ± 3.37	83.08 ± 2.63	82.54 ± 3.73
	Bayesian	88.47 ± 6.68	87.77 ± 3.76	88.27 ± 1.86	87.29 ± 1.38	86.04 ± 1.14

Table 5. Top-1 Accuracy on Different Datasets. Model MLP.

Dataset	Algorithm	Number of Clients 10	30	50	70	90
CIFAR10	FedTune	**40.57 ± 2.57**	**41.89 ± 1.17**	**42.51 ± 1.81**	**42.77 ± 1.15**	**43.62 ± 0.94**
	Random	36.02 ± 2.59	39.06 ± 1.45	40.61 ± 1.28	41.09 ± 1.24	41.99 ± 1.59
	Hyperband	36.56 ± 2.36	39.53 ± 1.96	40.38 ± 1.08	41.10 ± 1.73	41.24 ± 2.01
	Bayesian	38.78 ± 2.43	40.13 ± 1.43	40.73 ± 2.44	40.98 ± 1.24	40.36 ± 1.60
FMNIST	FedTune	**78.50 ± 3.49**	**78.19 ± 2.21**	**79.41 ± 1.27**	**78.63 ± 1.04**	**79.68 ± 0.86**
	Random	74.89 ± 3.78	75.85 ± 2.27	76.13 ± 2.24	77.01 ± 0.75	77.46 ± 1.04
	Hyperband	74.79 ± 5.48	75.15 ± 1.93	77.06 ± 1.78	77.52 ± 1.50	76.61 ± 1.77
	Bayesian	73.01 ± 2.65	74.44 ± 1.90	74.02 ± 1.22	75.77 ± 1.43	75.66 ± 1.19
MNIST	FedTune	**89.28 ± 3.33**	**89.62 ± 1.72**	**90.19 ± 1.18**	**89.90 ± 1.09**	**90.12 ± 0.81**
	Random	83.29 ± 3.70	84.78 ± 2.39	86.57 ± 1.60	85.47 ± 1.77	86.22 ± 0.75
	Hyperband	83.66 ± 5.87	84.59 ± 2.97	86.52 ± 2.30	86.23 ± 1.50	86.59 ± 1.87
	Bayesian	85.71 ± 4.28	86.34 ± 2.33	87.40 ± 1.37	87.59 ± 2.12	88.22 ± 1.42