FedPop: Federated Population-based Hyperparameter Tuning

Haokun Chen[1,2*], **Denis Krompaß**[2], **Jindong Gu**[3*], **Volker Tresp**[1,4]
[1] Ludwig Maximilian University of Munich, Munich, Germany
[2] Siemens Technology, Munich, Germany
[3] University of Oxford, Oxford, England
[4] Munich Center for Machine Learning, Munich, Germany
{haokun.chen, denis.krompass}@siemens.com,
jindong.gu@outlook.com, volker.tresp@lmu.de

Abstract

Federated Learning (FL) is a distributed machine learning (ML) paradigm, in which multiple clients collaboratively train ML models without centralizing their local data. Similar to conventional ML pipelines, the client local optimization and server aggregation procedure in FL are sensitive to the hyperparameter (HP) selection. Despite extensive research on tuning HPs for centralized ML, these methods yield suboptimal results when employed in FL. This is mainly because their "training-after-tuning" framework is unsuitable for FL with limited client computation power. While some approaches have been proposed for HP-Tuning in FL, they are limited to the HPs for client local updates. In this work, we propose a novel HP-tuning algorithm, called Federated Population-based Hyperparameter Tuning (FedPop), to address this vital yet challenging problem. FedPop employs population-based evolutionary algorithms to optimize the HPs, which accommodates various HP types at both the client and server sides. Compared with prior tuning methods, FedPop employs an online "tuning-while-training" framework, offering computational efficiency and enabling the exploration of a broader HP search space. Our empirical validation on the common FL benchmarks and complex real-world FL datasets, including full-sized Non-IID ImageNet-1K, demonstrates the effectiveness of the proposed method, which substantially outperforms the concurrent state-of-the-art HP-tuning methods in FL.

Introduction

Federated Learning (FL) is an effective machine learning paradigm suitable for decentralized data sources (McMahan et al. 2017). Similar to the conventional ML algorithms, FL exhibits sensitivity to empirical choices of hyperparameters (HPs), such as learning rate, and optimization steps (Kairouz et al. 2021). Hyperparameter Tuning (HPT) is a vital yet challenging component of the ML pipeline, which has been extensively studied in the context of centralized ML (Hutter, Kotthoff, and Vanschoren 2019). However, traditional HPT methods, such as Bayesian Optimization (Snoek, Larochelle, and Adams 2012), are not suitable for FL systems. These methods typically utilize the "training-after-tuning" framework. Within this framework, a substantial number of HPs needs to be evaluated, which involves repetitive training of models until convergence and subsequent retraining after optimizing the optimal HP. Such approaches can drastically increase the client's local computational costs and communication overheads, as it needs to execute multiple federated communications when evaluating only one HP. Furthermore, the distributed validation datasets impose a major challenge for HPT in FL, making it infeasible to evaluate HP for a large number of participating clients.

Recently, a few approaches have emerged to address the problem intersection of HPT and FL, but they still exhibit certain limitations: FedEx (Khodak et al. 2021) predefines a narrower HP search space, while FLoRA (Zhou et al. 2023) requires costly retraining after HP-optimization. Moreover, they are only applicable for tuning the HPs used in client local updates. In this paper, we propose Federated Population-based Hyperparameter Tuning (`FedPop`) to address the challenge of tuning HPs for FL. `FedPop` applies population-based evolutionary algorithm (Jaderberg et al. 2017) to optimize the HPs, which adds minimal computational overheads and accommodates various HP types at the client and server sides. Most importantly, `FedPop` employs an online "tuning-while-training" framework, enhancing efficiency and thereby allowing the exploration of a broader HP search space.

In `FedPop`, we first construct multiple HP-configurations as our tuning population, i.e., we initialize multiple tuning processes (members) with randomly initialized HP-configuration, containing the HPs used in the server aggregation and the local client updates. Afterwards, we apply an evolutionary update mechanism to optimize the HPs of each member by leveraging information across different HP-configurations (`FedPop-G`). Hereby, the HPs in underperforming members will be replaced by a perturbed version of the HPs from better-performing ones, enabling an efficient and effective online propagation of the HPs. To further improve the HPs for the local client updates in a fine-grained manner, we consider the active clients in each communication round as our local population, where each member contains one HP-vector used in the local client update (`FedPop-L`). Similarly, evolutionary updates are executed based on the local validation performance of each member to tune these HP-vectors. Most importantly, all the tuning processes, i.e., members of the population, are decentralized

*Corresponding author

and can be asynchronous, aligning perfectly with the distributed system design.

The proposed algorithm FedPop achieves new state-of-the-art (SOTA) results on three common FL benchmarks with both vision and language tasks, surpassing the concurrent SOTA HPT method for FL, i.e., FedEx (Khodak et al. 2021). Moreover, we evaluate FedPop on large-scale cross-silo FL benchmarks with feature distribution shift (Li et al. 2021), where its promising results demonstrate its applicability to complex real-world FL applications. Most importantly, we demonstrate the scalability of FedPop, where we show its applicability to full-sized ImageNet-1K (Deng et al. 2009) with ResNet-50 (He et al. 2016). Our contributions in this paper can be summarized as follows:

- We propose an effective and efficient online hyperparameter tuning (HPT) algorithm, FedPop, to address HPT problem for decentralized ML systems.
- We conduct comprehensive experiments on three common FL benchmarks with both vision and language tasks, in which FedPop achieves new SOTA results.
- We verify the maturity of FedPop for complex real-world cross-silo FL applications, and further analyze its convergence rate on full-sized non-IID ImageNet-1K, as well as its effectiveness when combined with various federated optimization algorithms.

Related Works

Hyperparameter Tuning for FL System

Previous works for tuning hyperparameters in FL focus only on specific aspects: (Wang et al. 2019) tunes only the local optimization epochs based on the client's resources, while (Koskela and Honkela 2018; Mostafa 2019; Reddi et al. 2020) focus on the learning rate of client local training. (Dai, Low, and Jaillet 2020, 2021) apply Bayesian Optimization (BO) (Snoek, Larochelle, and Adams 2012) in FL and optimize a personalized model for each client, while (Tarzanagh et al. 2022) computes federated hypergradient and applies bilevel optimization. (He, Annavaram, and Avestimehr 2020; Xu et al. 2020; Garg, Saha, and Dutta 2020; Seng et al. 2022; Khan et al. 2023) tune architectural hyperparameters, in particular, adapt Neural Architecture Search (NAS) for FL. (Zhang et al. 2022) tunes hyperparameter based on the federated system overheads, while (Maumela, Nelwamondo, and Marwala 2022) assumes the training data of each client is globally accessible. (Mlodozeniec, Reisser, and Louizos 2023) partitions both clients and the neural network and tunes only the hyperparameters used in data augmentation. (Khodak et al. 2020, 2021) systematically analyze the challenges of hyperparameter tuning in FL and propose FedEx for client local hyperparameters. (Zhou et al. 2023) proposes a hyperparameter optimization algorithm that aggregates the client's loss surfaces via single-shot upload. In contrast, the proposed method, FedPop, is applicable to various HP types on the client and server sides. In addition, it does not impose any restrictions on data volume and model architecture.

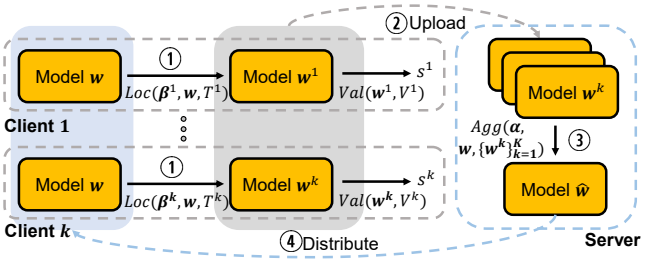

Figure 1: Schematic illustration of the operations involved in one communication round, summarized as Fed-Opt.

Evolutionary Algorithms

Evolutionary algorithms are inspired by the principles of natural evolution, where stochastic genetic operators, e.g., mutation and selection, are applied to the members of the existing population to improve their survival ability, i.e., quality (Telikani et al. 2021). Evolutionary algorithms have shown their potential to improve machine learning algorithms, including architecture search (Real et al. 2017; Liu et al. 2017), hyperparameter tuning (Jaderberg et al. 2017; Parker-Holder, Nguyen, and Roberts 2020), and Automated Machine Learning (AutoML) (Liang et al. 2019; Real et al. 2020). FedPop employs an online evolutionary algorithm, which is computationally efficient and explores a broader HP search space. To the best of our knowledge, FedPop is the first work combining evolutionary algorithms with HP optimization in Federated Learning.

Federated Hyperparmater Tuning

Problem Definition

In this section, we introduce the problem setup of hyperparameter tuning for FL. Following the setting introduced in (Khodak et al. 2021), we assume that there are $N_c \in \mathbb{N}^+$ clients joining the federated communication. Each client k owns a training, validation, and testing set, denoted by T^k, V^k, and E^k, respectively. To simulate the communication capacity of a real-world federated system, we presume that there are exactly $K \in \mathbb{N}^+$ active clients joining each communication round. In FedAvg (McMahan et al. 2017), the central server obtains the model weight $\boldsymbol{w} \in \mathbb{R}^d$ by iteratively distributing $\boldsymbol{w}$ to the active clients and averaging the returned optimized weights, i.e., $\{\boldsymbol{w}^k | 1 \leq k \leq K\}$.

More specifically, we denote the server aggregation and the client local training functions as Agg and Loc, respectively. Our goal is to tune the hyperparameter vectors (**HP-vectors**) used in these two functions. In particular, we denote the HP-vector used in Agg and Loc as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, which are sampled from the hyperparameter distribution H_a and H_b, respectively. We define the combination of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as one **HP-configuration**. In the following, we explain the general steps executed in the communication round, which involves these functions and HP-configurations. We summarize these steps as federated optimization (Fed-Opt), which is illustrated in Figure 1. Specifically, all active clients first execute

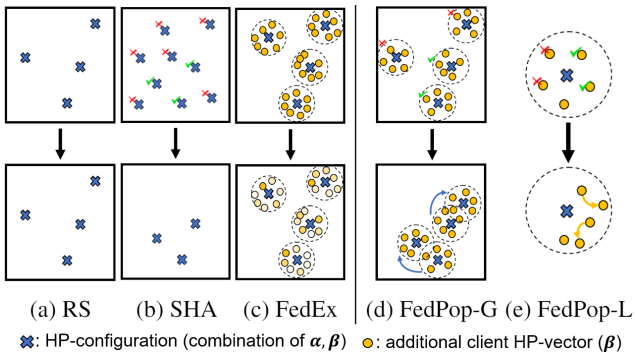

Figure 2: Schematic comparison between FedPop and other baselines. FedEx optimizes the sampling probabilities of additional $\boldsymbol{\beta}$ based on validation performance (indicated by the brightness of the yellow dots). In contrast, our method supports the tuning of both server (FedPop-G) and clients (FedPop-G and -L) HP-vectors and explores broader search space with the help of evolutionary updates.

function Loc (①) in parallel:

$$\boldsymbol{w}^k \leftarrow \texttt{Loc}(\boldsymbol{\beta}^k, \boldsymbol{w}, T^k), \tag{1}$$

which takes the HP-vector $\boldsymbol{\beta}^k$, model parameters $\boldsymbol{w}$ distributed by the central server, and the local training set T^k as inputs, and outputs the optimized model weight $\boldsymbol{w}^k$. Afterwards, the central server aggregates $\boldsymbol{w}^k$, uploaded by the active clients (②) , and executes function Agg (③):

$$\hat{\boldsymbol{w}} \leftarrow \texttt{Agg}(\boldsymbol{\alpha}, \boldsymbol{w}, \{\boldsymbol{w}^k | 1 \leq k \leq K\}), \tag{2}$$

which takes HP-vector $\boldsymbol{\alpha}$, current model parameter $\boldsymbol{w}$, updated model parameters from the active clients $\{\boldsymbol{w}^k | 1 \leq k \leq K\}$, and outputs the aggregated model weight $\hat{\boldsymbol{w}}$ which will be distributed to the active clients in the next communication round (④). The goal of the federated hyperparameter tuning method is to find the optimal HP-vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ within a predefined communication budget.

Challenges in Federated Hyperparameter Tuning

Given the problem defined in the previous section, we describe the two main challenges when tuning the hyperparameters for federated learning:

(C1) Extrem resource limitations: The communication budgets for optimizing ML models via FL are always very constrained due to the limited computational power of the clients and connection capacity of the overall system (Li et al. 2020). Therefore, common hyperparameter tuning algorithms, such as extensive local hyperparameter tuning for each client, or experimenting multiple hyperparameter configurations for the overall federated system and then retraining, may not be suitable in the context of FL.

(C2) Distributed validation data: In centralized ML, most hyperparameter tuning algorithms select the HP-configurations based on their validation performance. However, the validation data (V^k) is distributed across the clients in FL. Computing a validation score over all clients is extremely costly and thus infeasible for FL. The alternative is to use the validation performance of client subsets, e.g., the active clients of the communication round, which greatly reduces computational costs. However, this may lead to evaluation bias when the client data are not independent and identically distributed (*Non-IID*).

Method	Number of tried $\boldsymbol{\alpha}$	Number of tried $\boldsymbol{\beta}$	Optim. of $\boldsymbol{\alpha}$	Optim. of $\boldsymbol{\beta}$
RS	5	5	✗	✗
SHA	27	27	✗	✗
FedEx	5	135	✗	✓
FedPop	45	>1000	✓	✓

Table 1: Number of HP-vectors tested in different HP-tuning methods on CIFAR-10 benchmark. FedPop experiments the largest number of HP-configurations among all methods. Detailed computations are provided in the Appendix.

Baseline Methods

Before introducing the proposed algorithm (FedPop) which addresses the challenges of HP-tuning in FL, we illustrate the adaptation of two widely adopted HP-tuning baselines for FL applications and their notations. For the FL setup, we define the maximum communication rounds for the FL system, i.e., total tuning budget, as R_t, and the number of initial HP-configurations as N_c, respectively. We devise two baseline methods for tuning $\boldsymbol{\alpha}, \boldsymbol{\beta}$ as follows:

(1) **Random Search (RS)** first initializes N_c HP-configurations, resulting in a tuning budget of R_c $(= \frac{R_t}{N_c})$ for each HP-configuration. Afterwards, an ML model and N_c tuning processes will be initialized, where each tuning process executes R_c communication rounds to optimize the model using one specific HP-configuration. Finally, the optimized models from all tuning processes will be evaluated and the model exhibiting the highest testing accuracy, as well as its corresponding HP-configuration, are saved.

(2) **Successive Halving (SHA)** is a variation of RS which eliminates $\frac{1}{\eta}$-quantile of the under-performing HP-configurations after specific numbers of communication rounds. Within the same tuning budget R_t, SHA is able to experiment more HP-configurations compared with RS, thus increasing the likelihood of achieving better results. Based on R_t, N_c, and the number of elimination operations, the time step for elimination can be computed. However, the elimination might also discard HP-configurations which lead to promising results but perform poorly at early stages.

Limitations: These baseline methods exhibit two limitations when adapted to FL applications: First, as shown in Figure 2, their numbers of HP-configurations, as well as the HP values, are pre-defined and remain fixed throughout the tuning process. Second, these baseline methods are "static" and no active tuning is executed inside each tuning process. In other words, the model evaluation results are only obtained and utilized after a specific number of communication rounds. Therefore, we propose FedPop, a population-based tuning algorithm that updates the HP-configurations via evolutionary update algorithms. As a result of its high efficiency,

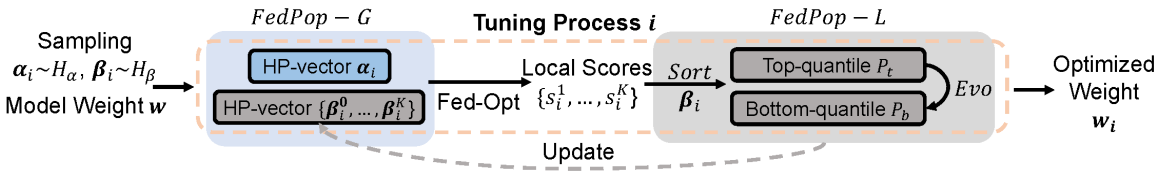

Figure 3: Schematic illustration of FedPop, including FedPop-L for intra-configuration HP-tuning and FedPop-G for inter-configuration HP-tuning. FedPop employs an online "tuning-while-training" schema for tuning both server ($\boldsymbol{\alpha}$) and clients ($\boldsymbol{\beta}$) HP-vectors. All functions in FedPop can be executed in a parallel and asynchronous manner. *Best viewed in color.*

it experiments the largest number of HP-vectors among all methods (Table 1), which is introduced in the following.

Proposed Method

A schematic illustration of the proposed method, Federated Population-Based Hyperparameter Tuning (FedPop), is provided in Figure 3. First, we randomly sample the HP-vectors ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) for each tuning process *in parallel* and execute federated optimization Fed-Opt (Figure 1). Subsequently, we conduct FedPop based on the validation scores s returned from the active clients in each tuning process. FedPop can be divided into 2 sub-procedures: FedPop-G aims at tuning both HP-vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ across all HP-configurations (*inter-config*), while FedPop-L focuses on a fine-grained search of HP-vector $\boldsymbol{\beta}$ inside each HP-configurations (*intra-config*).

FedPop can be wrapped with the aforementioned baselines. Specifically, the primary distinctions between the two wrappers are the number of initialized HP-configurations (N_c) and the execution of the tuning process eliminations in the intermediate steps. In the following, we use the most rudimentary method, RS, as our wrapper and elaborate on the proposed method. More details regarding FedPop wrapped with SHA are provided in the Appendix.

With RS as the wrapper, FedPop first randomly initializes N_c HP-configurations $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i^0)$ as the initial population and copies the model weight vector $\boldsymbol{w}$. Afterwards, we randomly sample addition K HP-vectors, i.e., $\{\boldsymbol{\beta}_i^k | 1 \leq k \leq K\}$, inside a small Δ-ball centered by $\boldsymbol{\beta}_i^0$. Δ is selected based on the distribution of the HP (H_b) and more details are provided in the Appendix. Directly sampling $\boldsymbol{\beta}_i^k$ from H_b is problematic because we find that using too distinct HP-vectors for the active clients would lead to unstable model performance. This phenomenon was also observed by (Khodak et al. 2021). We provide a schematic illustration of the sampling process in Figure 2, where the *yellow dots* ($\{\boldsymbol{\beta}_i^k | 1 \leq k \leq K\}$) are enforced to lie near the *blue crosses* ($\boldsymbol{\beta}_i^0$). Note that this resampling process of $\boldsymbol{\beta}_i^k$ is also executed when $\boldsymbol{\beta}_i^0$ is perturbed via Evo in FedPop-G. Finally, R_c communication rounds are executed for each tuning process in parallel, where the validation scores s_i^k, of the k_{th} active client in the i_{th} tuning process is recorded. The pseudo codes of the proposed method are given in Algorithm 1.

Evolution-based Hyperparameter Update (Evo): Inspired by Population-based Training (Jaderberg et al. 2017), we design our evolution-based hyperparameter update function Evo as the following,

$$\texttt{Evo}(\boldsymbol{h}) = \begin{cases} \hat{h}_j \sim U(h_j - \delta_j, h_j + \delta_j) \quad \text{s.t.} \quad H_j = U(a_j, b_j), \\ \hat{h}_j \sim U\{x_j^{i \pm \lfloor \delta_j \rceil}, x_j^i\} \quad \text{s.t.} \begin{cases} H_j = U\{x_j^0, ..., x_j^n\}, \\ h_j = x_j^i, \end{cases} \end{cases} \tag{3}$$

where $\boldsymbol{h}$ represents one HP-vector, i.e., $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ for our problem setting. We perturb the j_{th} value of $\boldsymbol{h}$, h_j, by resampling it from its possible neighboring values. Concretely, we select the new value of h_j based on the type of its original sampling distribution H_j: (1) If h_j is sampled from a continuous uniform distribution $H_j = U(a_j, b_j)$ (e.g., log-space of learning-rate, dropout), then we perturb h_j by resampling it from $U(h_j - \delta_j, h_j + \delta_j)$, where $\delta_j \leftarrow (b_j - a_j)\epsilon$ and ϵ is the pre-defined perturbation intensity. (2) If $h_j = x_j^i$ is sampled from a discrete uniform distribution $H_j = U\{x_j^0, ..., x_j^n\}$ (e.g., batch-size, epochs), then we perturb h_j by reselecting its value from $\{x_j^{i - \lfloor \delta_j \rceil}, x_j^i, x_j^{i + \lfloor \delta_j \rceil}\}$. To further increase the diversity of the HP search space during tuning, we resample h_j from its original distribution H_j with probability p_{re}.

While the HPs are randomly initialized in the early tuning stages, they become more informative as training progresses. To reflect this in FedPop, we employ a cosine annealing schema to control the values of ϵ and p_{re} based on the conducted communication rounds r:

$$x_r = \frac{x_0}{2} \cdot (1 + cos(\pi \frac{r}{R_c})), \tag{4}$$

where x_r and x_0 denote the present and the initial value of the annealed parameter, respectively, x is either ϵ or p_{re}.

FedPop-G for Inter-configuration Tuning: In FedPop-G, we adopt the average validation loss of all active clients, i.e., $s_i = \frac{1}{K} \sum_{k=1}^{K} s_i^k$, as the performance score for i_{th} HP-configuration. However, s_i may be a biased performance measurement, i.e., the disparity in the difficulty of the validation sets between different clients may lead to noisy s_i. To reduce the impact of the noise, FedPop-G is

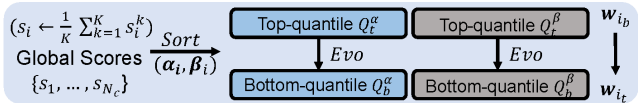

Figure 4: Schematic illustration of FedPop-G.

Algorithm 1: Federated Population-Based Hyperparameter Tuning.

Input: Number of active clients per round K, number of HP-configurations N_c, total communication budget R_t, communication budget for each HP-configuration R_c (computed by $\frac{R_t}{N_c}$), perturbation interval for `FedPop-G` T_g, initial model weight $\boldsymbol{w}$, N_c server HP-vectors $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_{N_c}\}$, N_c client HP-vectors $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1^0, ..., \boldsymbol{\beta}_{N_c}^0\}$.

Copy the model weights $\boldsymbol{w}_i \leftarrow \boldsymbol{w}$ for all N_c tuning processes.

for *comm. round* $r \leftarrow 1$ **to** R_c **do**

  for $i \leftarrow 1$ **to** N_c **do**

    `// in parallel`

    if $len(\boldsymbol{\beta}_i) == 1$ **then**

      Randomly sample $\{\boldsymbol{\beta}_i^k\}_{k=1}^K$ inside Δ-ball of $\boldsymbol{\beta}_i^0$.

    for *Client* $k \leftarrow 1$ **to** K **do**

      `// in parallel`

      $\boldsymbol{w}_i^k \leftarrow \texttt{Loc}(\boldsymbol{\beta}_i^k, \boldsymbol{w}_i, T^k)$

      $s_i^k \leftarrow \texttt{Val}(\boldsymbol{w}_i^k, V^k)$

    $\boldsymbol{\beta}_i \leftarrow \texttt{FedPop-L}\ (\boldsymbol{\beta}_i, \{s_i^k\}_{k=1}^K, K)$

    $\boldsymbol{w}_i \leftarrow \texttt{Agg}(\boldsymbol{\alpha}_i, \boldsymbol{w}_i, \{\boldsymbol{w}_i^k\})$

    $s_i \leftarrow \frac{1}{K}\sum_{k=1}^K s_i^k$

  if $r\%T_g = 0$ **then**

    $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{w}_i\}_{i=1}^{N_c} \leftarrow \texttt{FedPop-G}$
    $(\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{w}_i, s_i\}_{i=1}^{N_c}, N_c)$

return $\{\boldsymbol{w}_i\}_{i=1}^{N_c}$

Function *FedPop-L*$(\boldsymbol{\beta}, \boldsymbol{s}, K)$

  $\boldsymbol{P}_b \leftarrow \{k : s^k \geq \frac{\rho-1}{\rho}\text{-quantile}(\{s^k\})\}$

  $\boldsymbol{P}_t \leftarrow \{k : s^k \leq \frac{1}{\rho}\text{-quantile}(\{s^k\})\}$

  for $k_b \in \boldsymbol{P}_b$ **do**

    Sample k_t from $\boldsymbol{P}_t$.

    Delete $\boldsymbol{\beta}^{k_b}$.

    $\boldsymbol{\beta}^{k_b} \leftarrow \texttt{Evo}(\boldsymbol{\beta}^{k_t})$

  return $\boldsymbol{\beta}$

Function *FedPop-G*$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{s}, N_c)$

  $\boldsymbol{Q}_b \leftarrow \{i : s_i \geq \frac{\rho-1}{\rho}\text{-quantile}(\{s_i\})\}$

  $\boldsymbol{Q}_t \leftarrow \{i : s_i \leq \frac{1}{\rho}\text{-quantile}(\{s_i\})\}$

  for $i_b \in \boldsymbol{Q}_b$ **do**

    Sample i_t from $\boldsymbol{Q}_t$.

    Delete $\boldsymbol{\alpha}_{i_b}, \boldsymbol{\beta}_{i_b}, \boldsymbol{w}_{i_b}$.

    $\boldsymbol{\alpha}_{i_b}, \boldsymbol{\beta}_{i_b}^0 \leftarrow \texttt{Evo}(\boldsymbol{\alpha}_{i_t}, \boldsymbol{\beta}_{i_t}^0)$

    $\boldsymbol{w}_{i_b} \leftarrow \boldsymbol{w}_{i_t}$

  return $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}$

conducted with an interval of T_g communication rounds. Hereby, the list of scores s_i over T_g rounds are recorded and their weighted sum with a power-law weight decay (γ_g) is utilized as the final measurement:

$$s_i = \frac{\sum_{r=1}^{T_g} \gamma_g^{T_g - r} \cdot s_i^{(r)}}{\sum_{r=1}^{T_g} \gamma_g^{T_g - r}}. \tag{5}$$

The tuning procedure starts by sorting the HP-configurations according to their validation scores. Afterwards, 2 subsets, i.e., $\boldsymbol{Q}_b$ and $\boldsymbol{Q}_t$, are constructed, representing the indices of the bottom and top $\frac{1}{\rho}$-quantile of the HP-configurations, respectively. Finally, the HP-configurations with indices in $\boldsymbol{Q}_b$ will be replaced by the perturbed version of the HP-configurations with indices in $\boldsymbol{Q}_t$. Specifically, $\boldsymbol{\alpha}_{i_b}, \boldsymbol{\beta}_{i_b}^0$ are replaced by the perturbed version of $\boldsymbol{\alpha}_{i_t}, \boldsymbol{\beta}_{i_t}^0$ via `Evo` (Equation 3), the model weight in i_b-th HP-configuration ($\boldsymbol{w}_{i_b}$) are replaced by the i_t-th ($\boldsymbol{w}_{i_t}$).

`FedPop-L` for Intra-configuration Tuning: To further explore the local neighborhood of $\boldsymbol{\beta}_i^0$ for client local update in a fine-grained manner, we apply `FedPop-L` inside each tuning process. Hereby, we provide an informative assessment of $\boldsymbol{\beta}_i^0$ and its local neighborhood to enhance the robustness of HP-configuration. For simplicity, we omit i in the following notations. We consider the base HP-vector $\boldsymbol{\beta}^0$ as the perturbation center and restrict the perturbated HP-vector to lie inside a Δ-ball of it, i.e., $||\boldsymbol{\beta}^k - \boldsymbol{\beta}^0||_2 \leq \Delta$. At each communication round, $\boldsymbol{\beta}^k$ will be assigned to `Loc` of the k_{th} active client, the validation loss of the optimized model $\boldsymbol{w}^k$ will be recorded as the score s^k for HP-vector $\boldsymbol{\beta}^k$. Afterwards, $\{\boldsymbol{\beta}^k\}_{k=1}^K$ will be sorted according to the validation scores and separated into 2 subsets, containing the indices of the bottom ($\boldsymbol{P}_b$) and the top ($\boldsymbol{P}_t$) $\frac{1}{\rho}$-quantile of the $\boldsymbol{\beta}$, respectively. Finally, the HP-vectors $\boldsymbol{\beta}^{k_t}$ with indices in $\boldsymbol{P}_t$ will be perturbed to replace the HP-vectors $\boldsymbol{\beta}^{k_b}$ with indices in $\boldsymbol{P}_b$ via `Evo`.

Solutions to Challenges: (**C1**) `FedPop` does not require Bayesian Optimization (Zhou et al. 2023) or gradient-based hyperparameter optimization (Khodak et al. 2021), which saves the communication and computation costs. Besides, `FedPop` utilizes an *online* evolutionary method (`Evo`) to update the hyperparameters, i.e., not "training-after-tuning" but "tuning-while-training", which eliminates the need for "retraining" after finding a promising HP-configuration. Note that all procedures in `FedPop` can be conducted in a parallel and asynchronous manner. (**C2**) `FedPop-G` is conducted every T_g communication rounds to mitigate the noise depicted in the validation scores of HP-configurations. Besides, `FedPop-L` dynamically searches and evaluates the local neighborhood of $\boldsymbol{\beta}$, providing a more informative guidance for the client local HP optimization. Consequently, by enhancing the robustness of $\boldsymbol{\beta}$ to HP perturbation, we aim at improving its robustness against client data Non-IIDness.

Experiments and Analyses

We conduct an extensive empirical analysis to investigate the proposed method and its viability. Firstly, we compare `FedPop` with the SOTA and other baseline methods on three common FL benchmarks following (Khodak et al. 2021). Subsequently, we validate our approach by tuning hyperparameters for complex real-world cross-silo FL settings. Besides, we conduct an ablation study on `FedPop` to demonstrate the importance of its components. Moreover,

Tuning Wrapper	Tuning Algorithm	CIFAR-10			FEMNIST		Shakespeare	
		IID	NIID ($Dir_{1.0}$)	NIID ($Dir_{0.5}$)	IID	NIID	IID	NIID
RS	None	69.04 ±7.38 (65.28 ±5.83)	63.47 ±3.14 (60.51 ±8.03)	62.88 ±8.13 (60.65 ±7.37)	82.86 ±1.24 (83.76 ±3.56)	79.06 ±5.59 (83.09 ±2.64)	33.76 ±11.27 (31.19 ±10.18)	32.67 ±12.27 (31.32 ±9.92)
	FedEx	67.91 ±7.15 (64.21 ±7.84)	64.34 ±5.28 (62.97 ±7.27)	63.22 ±7.13 (61.92 ±8.06)	82.84 ±0.80 (82.57 ±3.25)	82.14 ±1.60 (84.03 ±2.48)	42.68 ±7.24 (41.22 ±6.34)	44.28 ±8.78 (46.69 ±7.39)
	FedPop	**71.18** ±4.68 (**68.01** ±3.42)	**68.25** ±5.03 (**65.74** ±3.97)	**67.01** ±4.98 (**65.24** ±3.97)	**84.33** ±1.41 (**85.99** ±1.62)	**83.21** ±2.08 (**85.48** ±1.48)	**44.30** ±3.37 (**44.46** ±3.53)	**47.28** ±3.47 (**50.25** ±3.87)
SHA	None	78.57 ±2.39 (75.93 ±4.96)	70.37 ±5.03 (67.83 ±4.41)	68.65 ±4.68 (65.58 ±8.10)	83.81 ±0.45 (85.52 ±1.63)	80.62 ±2.88 (87.64 ±0.64)	52.23 ±2.54 (49.06 ±5.98)	51.68 ±0.95 (48.83 ±3.12)
	FedEx	79.83 ±2.59 (77.04 ±1.45)	72.02 ±4.91 (70.81 ±4.65)	69.69 ±7.03 (67.02 ±7.65)	81.19 ±3.24 (85.69 ±1.91)	82.76 ±0.54 (86.79 ±2.89)	51.79 ±1.25 (51.89 ±1.30)	51.26 ±2.73 (51.01 ±3.36)
	FedPop	**81.47** ±1.24 (**78.96** ±0.87)	**76.42** ±3.04 (**75.03** ±2.56)	**74.88** ±2.06 (**72.41** ±1.87)	**84.33** ±0.57 (**86.84** ±0.98)	**83.26** ±0.86 (**88.33** ±0.79)	**53.48** ±0.57 (**52.66** ±1.91)	**53.07** ±0.97 (**52.79** ±0.36)

Table 2: Evaluation results of different hyperparameter tuning algorithms on three benchmark datasets. We report the *global* and locally *finetuned* (in the brackets) model performance with mean ±std from 5-trial runs. The best results are marked in **bold**.

we present convergence analysis of FedPop and its promising scalability by training ResNets from scratch on *full-sized* ImageNet-1K with Non-IID label distribution. Finally, we demonstrate the applicability of FedPop when combined with different federated optimization methods.

Benchmark Experiments

Datasets Description We conduct experiments on three benchmark datasets on both vision and language tasks: (1) *CIFAR-10* (Krizhevsky, Hinton et al. 2009), which is an image classification dataset containing 10 categories of real-world objects. (2) *FEMNIST* (Caldas et al. 2018), which includes gray-scale images of hand-written digits and English letters, producing a 62-way classification task. (3) *shakespeare* (Caldas et al. 2018) is a next-character prediction task and comprises sentences from Shakespeare's Dialogues.

We investigate 2 different partitions of the datasets: (1) For i.i.d (*IID*) setting, we randomly shuffle the dataset and evenly distribute the data to each client. (2) For non-i.i.d (*NIID*) settings, we follow (Khodak et al. 2021; Caldas et al. 2018) and assume each client contains data from a specific writer in FEMNIST, or it represents an actor in Shakespeare. For CIFAR-10 dataset, we follow prior arts (Zhu, Hong, and Zhou 2021; Lin et al. 2020) to model Non-IID label distributions using Dirichlet distribution Dir_x, in which a smaller x indicates higher data heterogeneity. We set the communication budget (R_t, R_c) to $(4000, 800)$ for CIFAR-10 and shakespeare, while $(2000, 200)$ for FEMNIST following (Khodak et al. 2021; Caldas et al. 2018). Besides, We adopt 500 clients for CIFAR-10, 3550 clients for FEMNIST, and 1129 clients for Shakespeare. For the coefficients used in FedPop, we set the initial perturbation intensity ϵ^0 to 0.1, the initial resampling probability p_{re}^0 to 0.1, and the quantile coefficient ρ to 3. The perturbation interval T_g for FedPop-G is set to $0.1R_c$. Following (Khodak et al. 2021), we define $\boldsymbol{\alpha} \in \mathbb{R}^3$ and $\boldsymbol{\beta} \in \mathbb{R}^7$, i.e., we tune learning rate, scheduler, and momentum for server-side aggregation (Agg), and learning rate, scheduler, momentum, weight-decay, the number of local epochs, batch-size, and dropout rate for local clients updates (Loc), respectively. More details about the HP search space, dataset descriptions, and model architectures are provided in Appendix.

Results and Discussion In Table 2, we report the testing accuracy achieved by the final model after performing hyperparameter tuning with different algorithms on three benchmarks. Hereby, we report the results of the *global* model, which is the server model $\boldsymbol{w}$ after the execution of the final communication round, and the *finetuned* model (in the brackets), which is the final global model finetuned on clients local data via $\texttt{Loc}(\boldsymbol{\beta}^0, \boldsymbol{w}, T^k)$. We observe that FedPop, combined with either RS or SHA as a wrapper, outperforms all the competitors on all benchmarks. For IID settings, the global model tuned on CIFAR-10 with FedPop, with RS or SHA as a wrapper, outperforms the baseline by 2.14% and 2.90%, respectively. Likewise, FedPop yields the highest average accuracy on FEMNIST and Shakespeare. For Non-IID settings, FedPop achieves a significant improvement of 3.85% and 4.79% on average compared with FedEx in CIFAR-10, when combined with RS and SHA, respectively. Moreover, we find that the performance improvement of the finetuned model using FedPop surpasses the other baselines. Additionally, we observe that during the tuning procedures, certain trials in the baselines and FedEx fail to converge. We attribute this to their predefined and fixed hyperparameter search spaces and values, resulting in higher sensitivity to the hyperparameter initialization that could not be mitigated during the tuning process. This phenomenon is observed via their larger accuracy deviation compared with FedPop, which further highlights the tuning stability of FedPop.

Validation on Real-World Cross-Silo FL Systems

As described in Section , previous hyperparameter tuning algorithms focused on small-scale benchmarks and simple model architectures. To indicate the effectiveness of FedPop on real-world FL applications, we further conduct experiments on three large-scale benchmarks: (1) PACS (Li et al. 2017), which includes images that belong to 7 classes from 4 domains Art-Painting, Cartoon, Photo, and Sketch.

Tuning Algorithm	PACS	OfficeHome	DomainNet
SHA	68.71 ±7.38	38.65 ±14.82	71.41 ±6.56
	(76.53 ±12.54)	(57.64 ±12.21)	(79.41 ±11.81)
FedEx	73.47 ±3.06	42.99 ±8.72	71.68 ±6.13
	(80.61 ±5.68)	(58.40 ±10.77)	(78.96 ±10.71)
FedPop	**75.17** ±1.18	**45.71** ±7.64	**73.59** ±3.58
	(**85.37** ±2.12)	(**62.76** ±7.38)	(**81.78** ±3.14)

Table 3: Evaluation results on three real-world cross-silo FL benchmarks with feature space distribution shifts.

(2) OfficeHome (Venkateswara et al. 2017), which contains 65 different real-world objects in 4 styles: Art, Clipart, Product, and Real. (3) DomainNet (Peng et al. 2019), which is collected under 6 different data sources: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch. All images are reshaped with larger sizes, i.e., 224x224. Following the setting proposed by (Li et al. 2021; Chen et al. 2023), we apply cross-silo (Li et al. 2020) FL settings and assume each client contains data from one of the sources (domains), but there exist feature distributions shift across different clients (feature space NIID (Li et al. 2021)). We use a more complex network architecture, i.e., ResNet-18, as the classification backbone. We set the tuning budget (R_t, R_c) to $(1000, 200)$. More details about the settings are provided in Appendix.

In Table 3, we report the evaluation results of the target model after tuning by `SHA` or its combination with `FedEx` or `FedPop`. We highlight the performance improvements achieved by the proposed method compared with the competitors, where `FedPop` surpasses the others up to 2.72% and indicates smaller accuracy deviations. These results indicate the effectiveness of `FedPop` on real-world FL scenarios with a smaller number of clients, large-scale private datasets, and more complex network architectures.

Ablation Study

To illustrate the importance of different `FedPop` components, we conduct an ablation study on CIFAR-10 benchmark considering *IID* and *NIID* settings. The results are shown in Table 4. We first notice that applying only one population-based tuning algorithm, i.e., either `FedPop-L` or `FedPoP-G`, already leads to distinct performance improvements on the baselines, especially when the client's

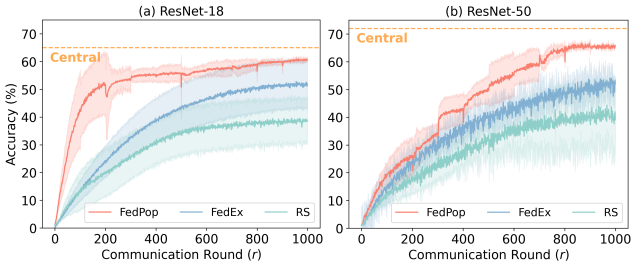

Figure 5: Convergence analysis of different tuning algorithms on full-sized *Non-IID* ImageNet-1k.

Tuning Algorithm	CIFAR-10		
	IID	NIID ($Dir_{1.0}$)	NIID ($Dir_{0.5}$)
RS	69.04 ±7.38	63.47 ±3.14	62.88 ±8.13
FedPop-G	70.61 ±3.21	66.81 ±3.24	65.63 ±4.67
FedPop-L	70.03 ±2.13	67.50 ±2.06	64.24 ±5.96
FedPop	**71.18** ±4.68	**68.25** ±5.03	**67.01** ±4.98

Table 4: Ablation study for different components in `FedPop` on CIFAR-10 benchmark.

data are *Non-IID*. Moreover, employing both functions together significantly improves the tuning results, which demonstrates their complementarity.

Analysis on Full-sized NIID ImageNet-1k

To further demonstrate the scalability of `FedPop`, we display the convergence analysis of `FedPop` on *full-sized* ImageNet-1K, where we distribute the data among 100 clients with Non-IID label distributions using Dirichlet distribution $Dir_{1.0}$. Hereby, we set $(R_t, R_c) = (5000, 1000)$ and report the average local testing results of the active clients after communication round r. We provide more details about the experimental setup in Appendix.

As shown in Figure 5, we discover that `FedPop` already outperforms the others from the initial phase, indicating its promising convergence rate. Besides, we also observe a reduced performance variation in `FedPop`, which further substantiates the benefits of evolutionary updates in stabilizing the overall tuning procedure. Most importantly, `FedPop` achieves comparable results with centralized training of the networks, indicating its scalability to tuning HPs for large-scale FL applications.

Conclusion and Outlooks

IIn this study, we introduce a novel population-based algorithm, `FedPop`, designed for hyperparameter tuning in distributed federated learning (FL) systems. Unlike conventional "training-after-tuning" approaches, `FedPop` adopts a "tuning-while-training" paradigm, making it uniquely suited for FL applications. The algorithm leverages evolutionary updates to optimize hyperparameters based on the performance of population members at both the client and server levels. Its global component `FedPop-G`, is applicable for tuning hyperparameters used in both server aggregation and client local updates. For a more detailed tuning of hyperparameters specific to client updates, we apply the fine-grained component, `FedPop-L`. Empirical results demonstrate that `FedPop-G` achieves state-of-the-art performance across three widely used FL benchmarks, handling both IID and non-IID data distributions. Furthermore, its promising performance on real-world FL tasks with feature distribution shifts underscores its effectiveness for complex applications. Finally, experiments on large-scale FL systems, including full-sized non-IID ImageNet-1K, validate its scalability and practical utility for real-world scenarios.

References

Caldas, S.; Duddu, S. M. K.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Chen, H.; Frikha, A.; Krompass, D.; Gu, J.; and Tresp, V. 2023. FRAug: Tackling federated learning with Non-IID features via representation augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4849–4859.

Dai, Z.; Low, B. K. H.; and Jaillet, P. 2020. Federated Bayesian optimization via Thompson sampling. *Advances in Neural Information Processing Systems*, 33: 9687–9699.

Dai, Z.; Low, B. K. H.; and Jaillet, P. 2021. Differentially private federated Bayesian optimization with distributed exploration. *Advances in Neural Information Processing Systems*, 34: 9125–9139.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Garg, A.; Saha, A. K.; and Dutta, D. 2020. Direct federated neural architecture search. *arXiv preprint arXiv:2010.06223*.

He, C.; Annavaram, M.; and Avestimehr, S. 2020. Towards non-iid and invisible data with fednas: federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210.

Khan, S.; Rizwan, A.; Khan, A. N.; Ali, M.; Ahmed, R.; and Kim, D. H. 2023. A multi-perspective revisit to the optimization methods of Neural Architecture Search and Hyper-parameter optimization for non-federated and federated learning environments. *Computers and Electrical Engineering*, 110: 108867.

Khodak, M.; Li, T.; Li, L.; Balcan, M.-F.; Smith, V.; and Talwalkar, A. 2020. Weight-Sharing for Hyperparameter Optimization in Federated Learning. In *Int. Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML*, volume 2020.

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

Koskela, A.; and Honkela, A. 2018. Learning rate adaptation for federated and differentially private learning. *arXiv preprint arXiv:1809.03832*.

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

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2017. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, 5542–5550.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60.

Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; and Dou, Q. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*.

Liang, J.; Meyerson, E.; Hodjat, B.; Fink, D.; Mutch, K.; and Miikkulainen, R. 2019. Evolutionary neural automl for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 401–409.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33: 2351–2363.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*.

Maumela, T.; Nelwamondo, F.; and Marwala, T. 2022. Population based training and federated learning frameworks for hyperparameter optimisation and ML unfairness using Ulimisana Optimisation Algorithm. *Information Sciences*, 612: 132–150.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Mlodozeniec, B.; Reisser, M.; and Louizos, C. 2023. Hyperparameter Optimization through Neural Network Partitioning. *arXiv preprint arXiv:2304.14766*.

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

Parker-Holder, J.; Nguyen, V.; and Roberts, S. J. 2020. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in Neural Information Processing Systems*, 33: 17200–17211.

Peng, X.; Bai, Q.; Xia, X.; Huang, Z.; Saenko, K.; and Wang, B. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, 1406–1415.

Real, E.; Liang, C.; So, D.; and Le, Q. 2020. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, 8007–8019. PMLR.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q. V.; and Kurakin, A. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, 2902–2911. PMLR.

Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; and McMahan, H. B. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Seng, J.; Prasad, P.; Dhami, D. S.; and Kersting, K. 2022. HANF: Hyperparameter And Neural Architecture Search in Federated Learning. *arXiv preprint arXiv:2206.12342*.

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

Tarzanagh, D. A.; Li, M.; Thrampoulidis, C.; and Oymak, S. 2022. FedNest: Federated bilevel, minimax, and compositional optimization. In *International Conference on Machine Learning*, 21146–21179. PMLR.

Telikani, A.; Tahmassebi, A.; Banzhaf, W.; and Gandomi, A. H. 2021. Evolutionary machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(8): 1–35.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5018–5027.

Wang, S.; Tuor, T.; Salonidis, T.; Leung, K. K.; Makaya, C.; He, T.; and Chan, K. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6): 1205–1221.

Xu, M.; Zhao, Y.; Bian, K.; Huang, G.; Mei, Q.; and Liu, X. 2020. Federated neural architecture search. *arXiv preprint arXiv:2002.06352*.

Zhang, H.; Zhang, M.; Liu, X.; Mohapatra, P.; and DeLucia, M. 2022. Fedtune: Automatic tuning of federated learning hyper-parameters from system perspective. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, 478–483. IEEE.

Zhou, Y.; Ram, P.; Salonidis, T.; Baracaldo, N.; Samulowitz, H.; and Ludwig, H. 2023. Single-shot general hyper-parameter optimization for federated learning. In *The Eleventh International Conference on Learning Representations*.

Zhu, Z.; Hong, J.; and Zhou, J. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, 12878–12889. PMLR.