Federated Hyperparameter Optimization through Reward-Based Strategies: Challenges and Insights

Krishna Kanth Nakka[1*] Ahmed Frikha[1] Ricardo Mendis[1]
Xue Jiang[1] Xuebing Zhou[1]
[1]Huawei Munich Research Center

Abstract

Performing hyperparameter tuning in federated learning is often prohibitively expensive due to the substantial communication overhead associated with training a single configuration, especially with a large hyperparameter search space. To overcome this challenge, recent works explored reward-based approaches to learn a policy distribution over a set of hyperparameter configurations. These approaches enable the concurrent exploration of multiple hyperparameter configurations within a single communication round, thereby accelerating the search process.

In this paper, we take a deeper look at the reward-based strategies and systematically analyze them, uncovering several issues and challenges associated with their adoption in practice. Furthermore, motivated by the insights from our analysis, we propose an in-depth evaluation of policy distribution with metrics that capture rankings of standalone configurations. We contribute this critical examination and proposed evaluation metrics in order to raise awareness about the challenges and hidden issues that reward-based federated hyperparameter optimization might face and to enable a more rigorous evaluation and therefore a faster progress in this research area. We expect that the identified challenges will serve as inspiration for the development of more robust and hyperparameter-free federated hyperparameter tuning approaches.

1. Introduction

Federated Learning (FL) [11] is a distributed training framework where a shared model is trained collaboratively on decentralized data. While FL has enjoyed a high interest across several industries, it still requires the careful and expensive selection of hyperparameters to achieve the best performance [4, 12, 20], making its adoption in real-world applications more challenging. While hyperparameter optimization (HPO) [5, 6, 21] has been extensively researched in the literature for centralized machine learning settings, these solutions are still prohibitively costly in the FL setting due to the large communication cost for training and, as a result, are not scalable to a large high-dimensional hyperparameter search space.

To address the challenging problem of federated hyperparameter optimization (FedHPO), researchers in [3, 10] proposed a framework well-suited to the decentralized setting. These approaches accelerate the search by several orders of magnitude, thanks to the collaborative setup that allows the concurrent exploration of hyperparameter configurations across different clients within a single communication round. In essence, they utilize a reward-based mechanism to learn an optimal policy distribution over the hyperparameter configuration. Despite the promising performance of these methods, a comprehensive understanding of them is lacking, limiting their adoption in real-world practical settings.

To this end, the present paper systematically examines reward-based federated hyperparameter tuning methods, identifies challenges they face, and proposes suitable evaluation metrics to assess the learned policy distribution and help uncover the identified issues. Our work has two main contributions. The first contribution consists of finding that reward-based hyperparameter tuning methods demand careful selection of internal hyperparameters inherent to the algorithm such as the discount factor, the learning rate scheduler, and the initial baseline to achieve optimal results. Further investigations uncover that the agent is biased towards the hyperparameter configurations sampled in the initial few rounds resulting in the under-exploration of the full configuration space. Notably, we also discover that there is a severe disparity in the rankings of the hyperparameter configurations derived from the policy distribution of the agent and the ground-truth rankings of standalone hyperparameter configurations. These observations highlight that reward-based hyperparameter tuning methods do not truly discover the top-performing configurations within the search space. Instead, they tend to rely on low-fidelity eval-

*Corresponding author. Email: krishkanth.92@gmail.com uations and are typically biased towards the configurations that are randomly sampled in the initial rounds, resulting in learning sub-optimal policy distribution.

Our second contribution consists of proposing the adoption of diverse ranking and correlation metrics to evaluate the policy distribution learned by the agent during the hyperparameter tuning process. We argue that prior works overlooked this evaluation aspect which is crucial for a rigorous assessment of the performance of the reward-based FedHPO algorithms. Our proposed metrics are inspired by the field of Neural Architecture Search [13, 16, 19, 28] which shares similarities with FedHPO. By evaluating the learned policy distribution with these metrics, we find that the rankings of configurations obtained from the policy distribution exhibit little (Kendall Tau ¡ 0.2) to no correlation with the rankings of standalone configurations. Our proposed evaluation metrics of policy distribution offer potential for designing more efficient tuning strategies, enabling the discovery of truly high-performing configurations. Finally, we believe that our comprehensive analysis of reward-based tuning methods will serve as inspiration for the development of more robust and hyperparameter-free federated hyperparameter tuning approaches in the future.

The remainder of the paper is structured as follows. Section 2 reviews existing work, focusing on approaches towards federated hyperparameter optimization. Section 3 presents the identified issues of existing reward-based FedHPO methods with details about the empirical analysis conducted. In Section 4, we introduce the evaluation metrics that we propose to understand the learned policy distribution. Finally, Section 5 concludes this work.

2. Background

Federated Learning (FL) [15, 27] and Hyperparameter Optimization (HPO) [2, 30] have been extensively studied as separate fields. However, the intersection of these two domains, known as Federated Hyperparameter Optimization (FedHPO), has received relatively limited attention. This is primarily due to the unique challenges posed by FL, particularly the high communication costs involved during training. While traditional efficient hyperparameter optimization techniques such as Successive Halving [5] can be applied to tune the hyperparameters in FL, the substantial communication cost for searching among a large number of configurations makes it a less favorable solution in practice.

To address these issues, recent works [3, 10] have introduced reward-based approaches to FedHPO, enabling *concurrent exploration* of multiple hyperparameter configurations within a single communication round. For example, Khodak et al. in [10] proposed FedEx for tuning the client-side hyperparameters by concurrently exploring multiple configurations across clients and learning the policy distribution over the hyperparameter configurations through a reward mechanism. Similarly, Cheng et al. in [3] introduced a deep learning-based policy network that outputs the personalized policy distribution taking encoded data features as input. At the core of these approaches lies a reward mechanism that learns a policy distribution over the hyperparameter configurations. In comparison to traditional HPO methods, the concurrent exploration of hyperparameter configurations makes them highly efficient and suitable for the collaborative FL paradigm. Apart from these, another recent work [24] conducts exploration using a single hyperparameter configuration at each communication round. However, this approach can incur a significant communication cost when dealing with a larger search space.

The intention of our paper is not to introduce a new hyperparameter tuning algorithm but rather to delve into the internal mechanisms of reward-based hyperparameter tuning methods. We believe that a deeper understanding of these mechanisms is crucial for addressing the challenging problem of hyperparameter optimization in the context of federated learning.

2.1. The FedEx Algorithm

In this paper, we conduct detailed studies on a representative framework, FedEx [10], to gain deeper insights on the reward-based tuning methods. We argue that FedEx, being a general-purpose reward-based tuning mechanism, makes it a favorable choice for our analysis. In broad terms, during each communication round t, for each client, the agent selects an action, which here corresponds to choosing a hyperparameter configuration from a set of available configurations, based on the current policy distribution, parametrized by $\boldsymbol{\theta}$. After training the model locally with the selected hyperparameter configuration, the reward is computed based on the validation loss at each client. This reward is then used to update the policy distribution $\boldsymbol{\theta}$. Ultimately, the agent seeks to learn an optimal distribution over the hyperparameter configurations that results in achieving high model performance.

The FedEx algorithm is detailed in Algorithm 1 and is summarized as follows. It starts in lines 1 and 2 by initializing the policy distribution $\boldsymbol{\theta_1}$ with uniform weights equal to $\frac{1}{k}$, server weights $\mathbf{w}_1$ and k hyperparameter configurations, $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, that are within a radius ϵ of each other. Then, for each communication round t, each client i receives both the current policy $\boldsymbol{\theta_t}$ and global weights $\mathbf{w}_t$ which are then used to perform local training (lines 3-7) with a configuration c_{ti} sampled from $\mathcal{D}_{\boldsymbol{\theta_t}}$ and the curret weights $\mathbf{w}_t$. After the local training, the updated local weights, sampled configuration, and respective model loss are sent back to the server, as illustrated in step 8. Finally, the server updates the policy $\boldsymbol{\theta_t}$ using an Exponential Gradient Descent (EGD) in lines 23 and 24, with a learning rate η_t that is dependent on the adopted scheduler scheme $\mathbb{S}$ (lines 11 to 22). After these

Algorithm 1 FedEx Algorithm Utilizing Reward-Based Mechanisms

```
Input: Sample single hyperparameter configuration $\mathbf{c}_k \in \mathbb{R}^k$ from original search space; number of clients $B$; number of communication rounds $\tau$. Hyperparameters: Discount factor $\gamma$; Scheduler scheme $\mathbb{S}$; learning rate $\eta$; initial baseline $\lambda_1$; configuration radius $\epsilon$
Output: Final model parameters $\mathbf{w}_t$ and hyperparameter distribution $\boldsymbol{\theta_t}$
 1: Initialize policy $\boldsymbol{\theta_1} \leftarrow [\frac{1}{k}, .., \frac{1}{k}]$ of dimension-$k$ and initial server model weights $\mathbf{w}_1 \in \mathbb{R}^d$
 2: Sample $k$ configurations $\{\mathbf{c}_2, \ldots, \mathbf{c}_k\}$ within configuration radius $\epsilon$ from $\mathbf{c}_1$
 3: for comm. round $t = 1, \ldots, \tau$ do
 4:     for client $i = 1, \ldots, B$ do
 5:         send $\mathbf{w}_t, \boldsymbol{\theta_t}$ to client
 6:         sample $c_{ti} \sim \mathcal{D}_{\boldsymbol{\theta_t}}$
 7:         $\mathbf{w}_{ti} \leftarrow \texttt{Loc}_{c_{ti}}(T_{ti}, \mathbf{w}_t)$
 8:         send $\mathbf{w}_{ti}, c_{ti}, L_{V_{ti}}(\mathbf{w}_{ti})$ to server
 9:     end for
10:     $\mathbf{w}_{t+1} \leftarrow \texttt{Agg}_b(\mathbf{w}, \{\mathbf{w}_{ti}\}_{i=1}^B)$
11:     if $t > 1$ then
12:         $\lambda_t \leftarrow \frac{1}{\sum_{s<t} \gamma^{t-s}} \sum_{s<t} \frac{\gamma^{t-s}}{\sum_{i=1}^B |V_{ti}|} \sum_{i=1}^B L_{V_{ti}}(\mathbf{w}_i)$
13:     end if
14:     $\tilde{\nabla}_j \leftarrow \frac{\sum_{i=1}^B |V_{ti}|(L_{V_{ti}}(\mathbf{w}_{ti}) - \lambda_t) 1_{c_{ti}=c_j}}{\boldsymbol{\theta_t}_{[j]} \sum_{i=1}^B |V_{ti}|} \ \forall\, j$
15:     $s \leftarrow \sum_{i=1}^B |V_{ti}| L_{V_{ti}} / \sum_{i=1}^B |V_{ti}|$
16:     if $\mathbb{S}$ == "constant" then
17:         $\eta_t = \sqrt{2 \log k}$
18:     else if $\mathbb{S}$ == "adaptive" then
19:         $\eta_t = \sqrt{2 \log k} / \sqrt{\sum_{s \leq t} \|\tilde{\nabla}_s\|_\infty^2}$
20:     else if $\mathbb{S}$ == "aggressive" then
21:         $\eta_t = \sqrt{2 \log k} / \|\tilde{\nabla}_t\|_\infty$.
22:     end if
23:     $\boldsymbol{\theta_{t+1}} \leftarrow \boldsymbol{\theta_t} \odot \exp(-\eta_t \tilde{\nabla})$
24:     $\boldsymbol{\theta_{t+1}} \leftarrow \boldsymbol{\theta_{t+1}} / \|\boldsymbol{\theta_{t+1}}\|_1$
25: end for
26: return model $\mathbf{w}_t$, hyperparameter distribution $\boldsymbol{\theta_t}$
```

updates, the algorithm returns to line 3 for the next round. After a predefined number of rounds, the algorithm returns the final global model weights $\mathbf{w}_t$ and hyperparameter distribution $\boldsymbol{\theta_t}$.

Notably, the algorithm relies on several internal hyperparameters such as the discount factor γ, the learning rate η_t, its scheduler $\mathbb{S}$, and the initial baseline λ_1. The discount factor γ controls the aggressiveness of the policy update through the baseline λ_t [18]. The learning rate η_t can be updated through multiple schemes such as constant, adaptive, and aggressive, proposed in the original framework. In short, the constant learning rate scheduler uses a fixed value for η_t for all the training rounds. The adaptive and aggressive learning rate schemes rely on gradient information from previous and current rounds to update the learning rate. Moreover, we emphasize that these hyperparameters are not specific to the current setting but are also prevalent in traditional reinforcement learning literature. Nevertheless, the addition of these hyperparameters incurs an additional burden, and should thus be robust. As such, several studies in the literature have sought to investigate the sensitivity of the hyperparameters involved in RL algorithms [1, 8, 18] and provided evidence that the RL algorithms are heavily influenced by several hyperparameters such as initial policy distribution, reward scale, learning rate, etc.

3. Experimental Insights

In this section, we will begin by examining the sensitivity of various hyperparameters within the context of the reward-based hyperparameter tuning algorithm, FedEx. We will then seek to understand the dynamics of the learning process for the reward policy across various settings. While our primary experimental results are centered around FedEx, it is important to note that the insights drawn from this analysis can be applied more broadly to reward-based hyperparameter tuning methods.

Implementation. We conduct our experiments using the CIFAR-10 dataset [14], which we split into B clients in an i.i.d. manner, with number of clients B set to either 4 or 30. We sample 27 hyperparameter configurations with a radius of $\epsilon = 1.0$, ie., from the entire search space. This means that with $B = 4$, we can sample up to 4 configurations out of the 27 to simulate the setting where the action space is large compared to exploration space, while with $B = 30$, there are more configurations than clients, allowing for extensive exploration. Following FedEx [10] implementation, we set the initial baseline λ_1 to 0.0. Further, we set the discount factor γ to 0.0 in all experiments unless otherwise stated. Since the original implementation tunes γ along with the server-side hyperparameter, we also perform ablation and tune γ in a limited set of experiments. The number of communication rounds is set to 200, and we sample all clients in each round to eliminate potential high variances in the observed validation loss due to different client samplings in each round. We use FedAvg for aggregating the client models at the server in each round. We tune 4 hyperparameters that are shared at all clients: dropout-rate [25] uniformly in the range $[0.0, 0.5]$; learning rate in log-uniform space $[10^{-4}, 10^{-1}]$; the momentum of the SGD optimizer uniformly in the range $[0.0, 1.0]$; the weight-decay coefficient in the log-uniform space $[10^{-5}, 10^{-1}]$. $\mathbf{c}_k$ is a hyperparameter configuration of dimension four containing hyperparameter values $\mathbf{c}_k^j$, where $j \in \{0, 1, 2, 3\}$, sampled from their respective search spaces.

Impact of the LR Scheduling Scheme $\mathbb{S}$

To investigate the impact of the learning rate (LR) scheduling scheme $\mathbb{S}$, we conducted experiments using the FedEx algorithm with two different total numbers of clients C: 4 and 30. As illustrated in Figure 1, we observed divergent trends with varying client sizes. When $C = 4$, we found that the *constant* scheduling scheme outperformed the other two schemes. However, with $C = 30$, we observed that the *aggressive* scheduling scheme yielded the best results. This shows the sensitivity of FedEx's performance to the choice of the scheduling scheme w.r.t. the experiment setting (such as the number of clients), suggesting that tuning the LR scheduling scheme $\mathbb{S}$ is crucial for achieving optimal results. We draw further insights regarding agent exploration and the impact of different LR scheduling schemes on the policy training later in this section.

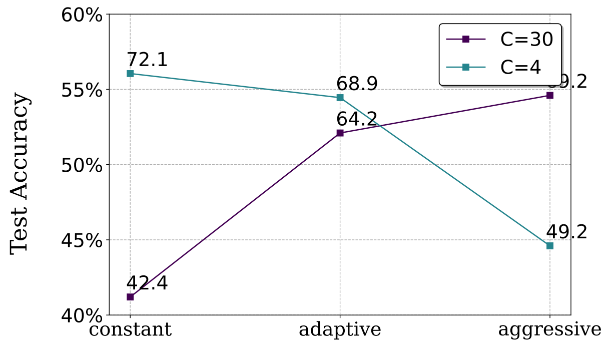


Figure 1. **Performance of Different Scheduling Schemes with Varying Numbers of Clients.** We observed that the performance of the scheduler is inconsistent across different client size settings. When $C = 4$, the constant scheduler outperforms other schemes, whereas with $C = 30$, the aggressive scheduler yields the best results. In this ablation, we set the discount factor $\gamma = 0.0$.

Impact of the Discount Factor γ

The discount factor γ controls the baseline updates, a crucial parameter for addressing the high variance of rewards in reinforcement learning frameworks [7, 18]. For example, $\gamma = 1.0$ sets the baseline as the average of all previous rewards, while $\gamma = 0.0$ takes the previous round reward as the baseline. In Figure 2, we illustrate how the discount factor affects the results under various scheduling schemes. Notably, we observe that $\gamma = 1.0$, $\gamma = 0.0$, and $\gamma = 0.5$ yield the best performance for the *aggressive*, *adaptive*, and *constant* scheduling schemes, respectively. This observation highlights the dependency of the results yielded by the FedEx algorithm on the discount factor γ, emphasizing the need for hyperparameter tuning within FedEx to achieve optimal performance.

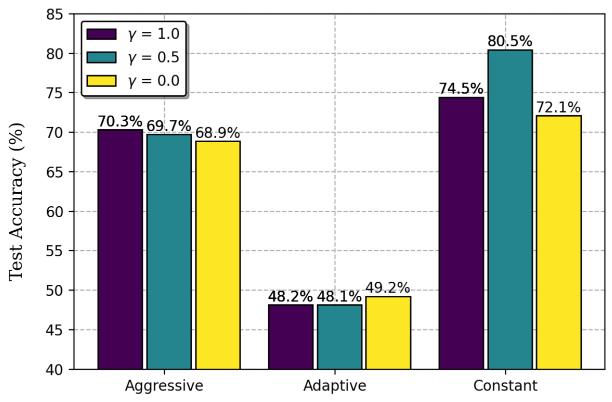


Figure 2. **Performance of Different Scheduling Schemes with Varying Reward Discount Factor γ.** We observe that the discount factor affects algorithm performance inconsistently across various scheduling schemes. In this setting, we fixed the number of clients at $C = 4$.

Impact of the Initialization of the Baseline λ_1

A crucial (hyper)parameter in reward-based tuning is the initial baseline λ_1. In each round, the baseline λ_1 is calculated based on previously observed rewards and the discount factor γ (step 12 in Algorithm 1). In the first round, the baseline λ_1 is conventionally set to 0, as in the FedEx implementation. We find that this initialization choice leads to unintended consequences, as it pushes the policy to update in the opposite direction. We illustrate this with a concrete case in the following.

Let's consider a specific client i at $t = 1$, i.e., first training round, and assume that we measured a validation loss $L_{V_{1i}}(\mathbf{w}_1)$ of 3.0 before local training with configuration $\mathbf{c}_{1i}$. After training with $\mathbf{c}_{1i}$, the validation loss $L_{V_{1i}}(\mathbf{w}_{1i})$ drops to 2.7, indicating that the used configuration yields a performance improvement. However, this still results in a *negative* validation performance gain of -2.7, computed as $(L_{V_{1i}}(\mathbf{w}_{ti}) - \lambda_1)$. Essentially, since the reward becomes negative (unless the $L_{V_{1i}}(\mathbf{w}_{1i})$ becomes smaller than $\lambda_1 = 0.0$ in the first round), the policy weights for *all* the sampled configurations $\{\mathbf{c}_{11}, .., \mathbf{c}_{1B}\}$, from the first round undergo substantial reduction in their weights. More importantly, non-sampled configurations have their policy weights increased as a result of penalizing the sampled configurations since $\boldsymbol{\theta}$ is a probability distribution and is therefore normalized at the end of each communication round t to ensure that the sum of elements remains 1. Consequently, the action/configuration sampling in the subsequent rounds becomes more biased towards configurations that are not sampled in the first round.

In our experiments, we addressed this issue by setting the initial baseline λ_1 to the average of the validation performance $\sum_{i=1}^{B} L_{V_{1i}}(\mathbf{w}_1)$ before starting the training. This adjustment yielded notable improvements of 4.3% for B=4 and 5.5% for B=30 in the final performance with the *adaptive* scheduling scheme. One could explore alternative approaches, such as skipping the policy update in the first (few) round(s). However, it's important to emphasize that even a simple change in the baseline of the first round significantly impacts the search algorithm's performance and final results. This observation underscores the sensitivity of the reward-based algorithms to the initialization of the baseline.

Understanding Agent Exploration

In the following, we explore how the policy $\boldsymbol{\theta}$ evolves with different LR scheduling schemes $\mathbb{S}$. In Figure 3, we examine the exploration of the agent with $B = 4$ using the *constant*, *adaptive*, and *aggressive* schemes in the three rows, respectively. The first column visualizes the ground-truth performance of all 27 configurations, ranging from 17% to 81%. The second column displays how many times each configuration is explored over the whole training procedure. In the third column, we plot the maximum value in the agent's policy θ throughout the training rounds to gain insights into the distribution of the policy weights.

Based on the results of our experiments with the number of clients $B = 4$ (Figure 3), we draw the following three conclusions. Firstly, the agent tends to underexplore the action space in the *aggressive* scheduling scheme. Similar under-exploration is observed with the *constant* scheduling scheme. Secondly, the agent's policy θ_i quickly converges to a deterministic case where a single configuration is sampled with a probability of 100%. This is observed within a few rounds (< 10) in the *aggressive* LR scheduling scheme and after approximately 70 rounds with the *constant* scheduling scheme. Thirdly and most importantly, the frequency of exploration of a configuration in all schemes does not correlate with the ground-truth performance of the configuration. For instance, the configuration with ID 4 is the best action according to the ground truth, yet it is hardly explored with any scheduling scheme.

To investigate whether the aforementioned issues arise due to the fact that the agent could only explore up to 4 unique configurations out of 27 in each round, we conduct an additional experiment with the number of clients $B = 30$. The results are presented in Figure 4. While setting the number of clients to a value higher than the number of configurations is expected to yield a sufficient exploration of the 27 configurations, we observe the same issues as in the experiment with $B = 4$. Namely, the under-exploration of actions with constant and aggressive schemes, the quick convergence of $\boldsymbol{\theta}$ to a deterministic sampling, and a significant disparity between ground truth and sampled frequency of a configuration.

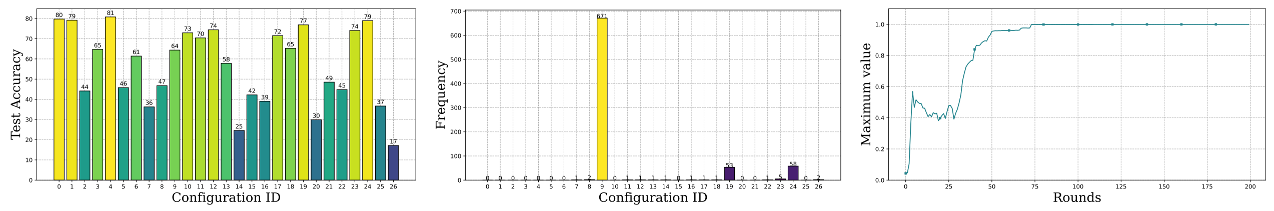
This experiment highlights that although *aggressive* scheduling is preferred in the original implementation for efficiency, the agent does not sufficiently explore the action space in this case, and the learned policy is heavily biased towards actions taken in the first few rounds. In contrast, the *adaptive* scheduling scheme allows for relatively better exploration. However, it still does not truly correlate with the ground-truth performance. Finally, *constant* scheduling scheme explores very few configurations before saturating to a single configuration.

4. Proposed evaluation metrics for FedHPO methods

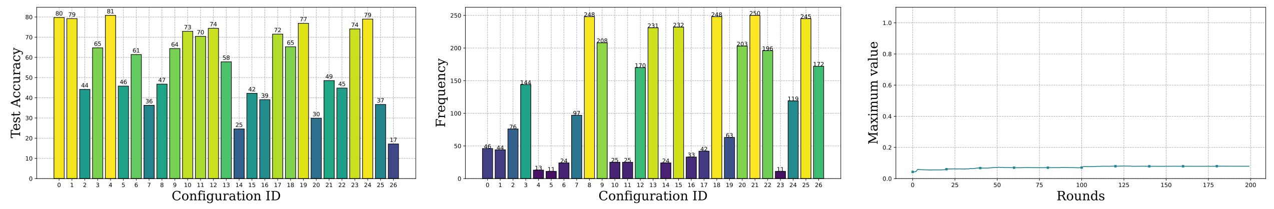
In this section, we propose evaluation metrics suitable for FedHPO approaches that contribute to uncovering the issues identified in Section 3 early in the method development process. While aiming to achieve higher model performance with $\mathbf{w}_t$ in terms of test set accuracy is the primary goal, we propose to evaluate the policy distribution $\boldsymbol{\theta_t}$ using correlation and ranking metrics. This proposal is inspired by the extensive literature in the domain of Neural Architecture Search (NAS) [13, 16, 19, 28], where architectures sampled from a trained super-net are evaluated based on their rankings compared to ground-truth standalone rankings. We strongly believe that a wide usage of these evaluation metrics across the FedHPO research community would accelerate the progress in this area.

We argue that achieving the best test accuracy should not be the sole goal when performing a hyperparameter search. Having a policy distribution that correlates well with standalone performances, i.e., reflects the true performance of each hyperparameter configuration, can be useful in various further applications. For instance, the policy distribution can be used as a good starting point to initialize the policy distribution of other datasets or tasks [22]. Moreover, it is possible to further fine-tune the model for additional rounds with configurations sampled from the learned distribution $\boldsymbol{\theta_t}$, which is common in NAS literature [17]. We hypothesize that this might be especially useful in the case of a data distribution shift, where the optimal hyperparameter configuration before the shift might become sub-optimal after the shift.

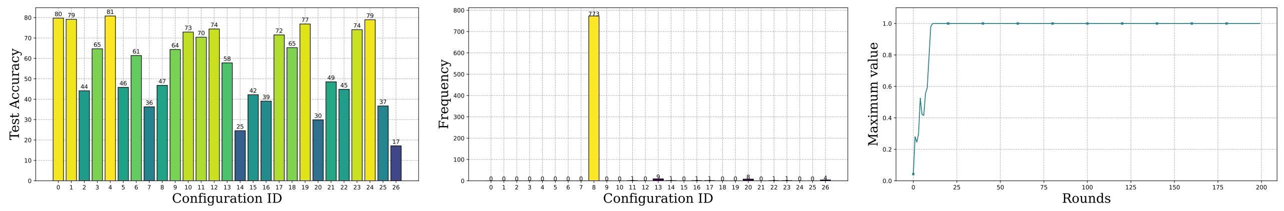
Metrics. We propose using three metrics: Kendall Tau [9], Spearman rank correlation [32], and $AP_n@k$ metric [19] to better assess the learned policy distribution. Using Kendall Tau allows measuring the correlation through pairwise rankings between the ground-truth *ranking* of the hyperparameter configurations, i.e., the ranking based on the performance achieved by each configuration when it is used separately, and the *ranking* of the configurations according to final policy distribution yielded by the FedHPO algorithm. We also propose to use the Spearman rank, to calculate the rank correlation coefficient between the ground-



(a) Constant scheduler



(b) Adaptive scheduler



(c) Aggressive scheduler

Figure 3. **Illustrating the Agent Learning Process with Different Scheduling Schemes with $B = 4$.** The figures from left to right plot the ground truth performance, the frequency of exploration and the maximum value in policy distribution $\boldsymbol{\theta}$.

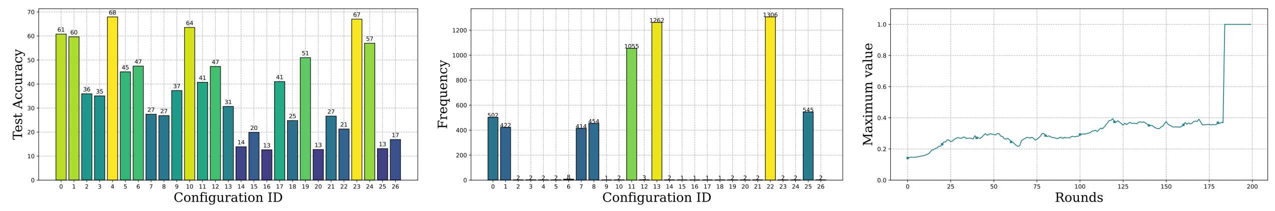
truth ranking of configurations and the policy vector $\boldsymbol{\theta_t}$. Furthermore, we propose to evaluate the policy using the Average Precision ($AP_n@k$) metric, a key metric in recommendation literature [23, 26], to measure how much the top elements in the ground truth and predicted configuration rankings match. Specifically, this metric compares the *top-n* configurations in the ground-truth ranking with the *top-k* configurations predicted by the policy distribution in terms of relative ranking and quantity. While the previous metrics give equal importance to all configurations, the $AP_n@k$ metric focuses on the *top-n* configurations. We set $n = 4$ and $k = 10$ for our evaluation.

Validation of the proposed metrics. In this section, we assess the ability of the proposed metrics in reflecting the issues of the reward-based FedHPO methods identified in Section 3. We do this by empirically evaluating the policy distribution $\boldsymbol{\theta_t}$ learned by the FedEx algorithm using our proposed metrics.

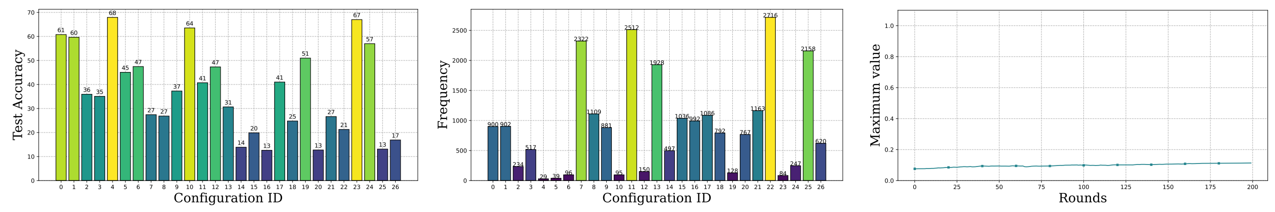
Our metrics indicate that the learned policy distribution $\boldsymbol{\theta_t}$ fails to capture the performance of standalone configurations, i.e., experiments where a single hyperparameter configuration is used, in most cases. In Figure 5, we visualize the metrics results. The columns present the three different LR scheduling schemes $\mathbb{S}$ and the rows display the results of the different client sizes, $B = 4$ and $B = 30$, respectively. The two correlation metrics lie in the range $[-1, 1]$, indicating a positive or negative correlation with higher absolute values showcasing a stronger correlation, while the AP ranges from 0 up to 1, with 1 being the highest performance.

We observe that both correlation metrics present values below 0.4 in all cases. In contrast, the $AP_4@10$ metric with *aggressive* scheduling and $B = 30$ (bottom-right subfigure) achieves a value over 0.6. However, this metric quickly converges to this value in less than 10 rounds and then remains constant for the rest of the training. Unsurprisingly, the other two metrics, Spearman and Kendall Tau, are still unsatisfactory in this case. Apart from this, we observe a slow drop in all metrics in the *adaptive* scheduling case as the training progresses, reaching values lower than -0.2, indicating a significant mismatch between the original rankings and predicted rankings. Note that negative correlation metric values indicate that the policy has learned a configuration ranking that partly has a reverse order.

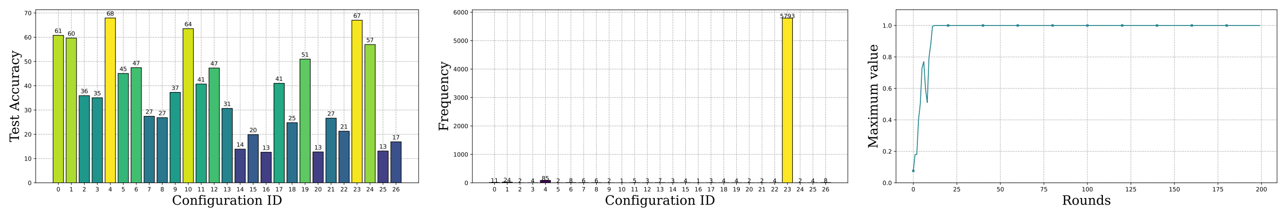
Overall, the evaluation metrics across the board suggest that the policy distribution learned by the agent does not capture the ground-truth ranking of the hyperparameter configurations, nor discover the top-performing ones. It is important to note that our primary aim is not to disprove reward-based schemes but rather to raise awareness about the issues they might encounter and to propose evaluation



(a) Constant scheduler

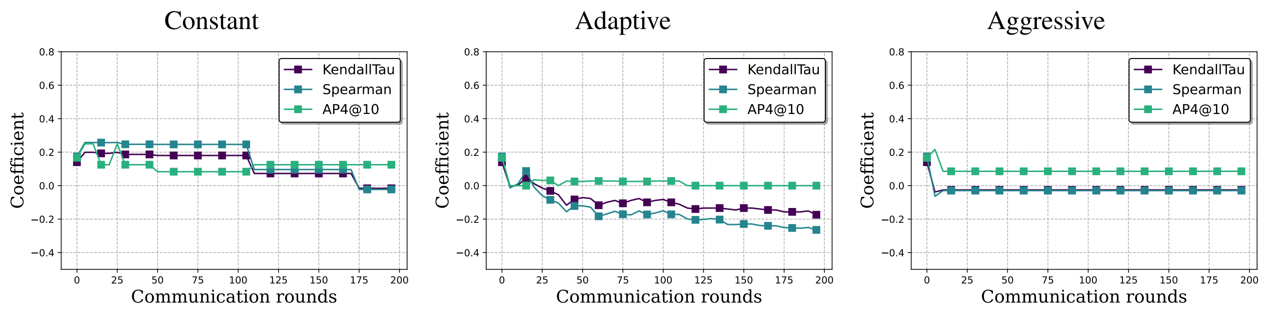


(b) Adaptive scheduler

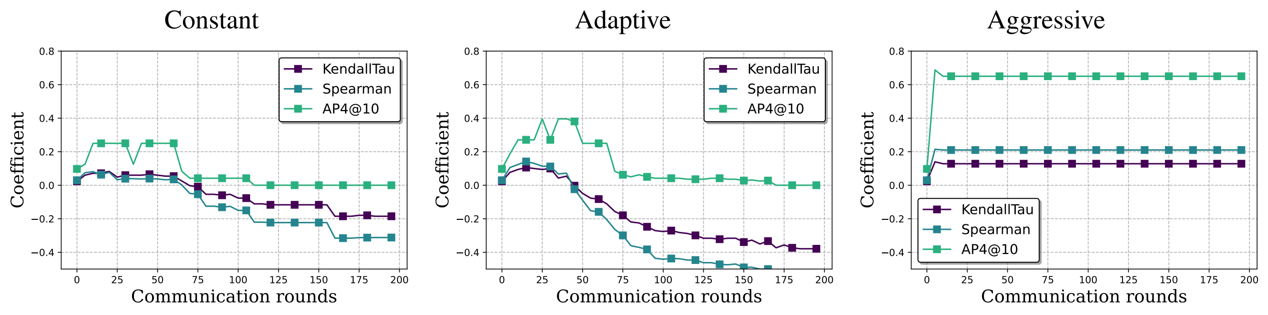


(c) Aggressive scheduler

Figure 4. **Illustrating the Agent Learning Process with Different Scheduling Schemes with $B = 30$.** The figures from left to right plot the ground truth performance, the frequency of exploration and the maximum value in θ.



(a) B=4



(b) B=30

Figure 5. **Evaluation of Policy distribution θ_t with different scheduling schemes.** We visualize the metric coefficients as the training progresses for three different scheduling schemes in three columns, respectively. The results are presented with B=4 and B=30 in the first and second row.

metrics to identify these issues.

Nonetheless, our observations suggest several promising directions for enhancing the learning of a more effective policy distribution. For instance, we could consider the incorporation of maximum entropy regularization [33] to encourage deeper exploration throughout the training rounds, and perform action elimination [31] to remove weak-performing configurations, or use a limited number of landmark configurations to guide the training [29]. Nevertheless, these techniques come with additional hyperparameters associated with their respective loss functions and thus require additional tuning.

5. Conclusion

In this paper, we have conducted a systematic study of reward-based federated hyperparameter tuning methods, using FedEx as a representative example. Our analysis has revealed the sensitivity of several internal hyperparameters within reward-based strategies, highlighting the importance of careful tuning to achieve optimal performance on the target task. This transfers the burden of hyperparameter tuning from the original task to tuning the search algorithm. Furthermore, a deeper examination of reward schemes yielded the following insights. Firstly, the agent tends to underexplore the action spaces in multiple settings. Secondly, the learned policy quickly converges to a deterministic policy that predicts the same single configuration. Moreover, we have observed significant inconsistencies in the rankings of configurations learned by the agent when compared to the ground-truth performance ranking of these configurations. Therefore, we propose evaluation metrics suitable to assess the policy distribution learned by the agent. In conclusion, we hope that our work will inspire further in-depth research in the field of federated hyperparameter tuning, ultimately leading to more robust, effective and hyperparameter-free methods.

References

[1] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020. 3

[2] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484, 2023. 2

[3] Anda Cheng, Zhen Wang, Yaliang Li, and Jian Cheng. Hpn: Personalized federated hyperparameter optimization. *arXiv preprint arXiv:2304.05195*, 2023. 1, 2

[4] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33:9687–9699, 2020. 1

[5] Matthias Feurer and Frank Hutter. Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33, 2019. 1, 2

[6] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. 1

[7] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004. 4

[8] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 3

[9] Maurice Kendall. A new measure of rank correlation. *Biometrika*, 1938. 5

[10] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina F Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *Advances in Neural Information Processing Systems*, 34:19184–19197, 2021. 1, 2, 3

[11] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016. 1

[12] Antti Koskela and Antti Honkela. Learning rate adaptation for federated and differentially private learning. *arXiv preprint arXiv:1809.03832*, 2018. 1

[13] Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nas-bench-suite-zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022. 2, 5

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3

[15] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 2

[16] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018. 2, 5

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 5

[18] Jincheng Mei, Wesley Chung, Valentin Thomas, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. The role of baselines in policy gradient optimization. *Advances in Neural Information Processing Systems*, 35:17818–17830, 2022. 3, 4

[19] Gaurav Mittal, Chang Liu, Nikolaos Karianakis, Victor Fragoso, Mei Chen, and Yun Fu. Hyperstar: Task-aware hyperparameters for deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8736–8745, 2020. 2, 5

[20] Hesham Mostafa. Robust federated learning through representation matching and adaptive hyper-parameters. *arXiv preprint arXiv:1912.13075*, 2019. 1

[21] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in neural information processing systems*, 33:17200–17211, 2020. 1

[22] Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021. 5

[23] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, page 53, 2011. 6

[24] Jonas Seng, Pooja Prasad, Devendra Singh Dhami, and Kristian Kersting. Hanf: Hyperparameter and neural architecture search in federated learning. *arXiv preprint arXiv:2206.12342*, 2022. 2

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3

[26] Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev. Quality metrics in recommender systems: Do we calculate metrics consistently? In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 708–713, 2021. 6

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

[28] Kaicheng Yu, Rene Ranftl, and Mathieu Salzmann. How to train your super-net: An analysis of training heuristics in weight-sharing nas. *arXiv preprint arXiv:2003.04276*, 2020. 2, 5

[29] Kaicheng Yu, Rene Ranftl, and Mathieu Salzmann. Landmark regularization: Ranking guided super-net training in neural architecture search. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 13723–13732, 2021. 8

[30] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020. 2

[31] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 8

[32] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of biostatistics*, 7, 2005. 5

[33] Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7553–7562. PMLR, 2019. 8