FEDHYPER: A UNIVERSAL AND ROBUST LEARNING RATE SCHEDULER FOR FEDERATED LEARNING WITH HYPERGRADIENT DESCENT

Ziyao Wang[1], Jianyu Wang[2], Ang Li[1]
[1]University of Maryland, College Park, [2]Apple
{ziyaow,angliece}@umd.edu, {jianyuwang}@apple.com

ABSTRACT

The theoretical landscape of federated learning (FL) undergoes rapid evolution, but its practical application encounters a series of intricate challenges, and hyperparameter optimization is one of these critical challenges. Amongst the diverse adjustments in hyperparameters, the adaptation of the learning rate emerges as a crucial component, holding the promise of significantly enhancing the efficacy of FL systems. In response to this critical need, this paper presents FEDHYPER, a novel hypergradient-based learning rate scheduling algorithm for FL. FEDHYPER serves as a universal learning rate scheduler that can adapt both global and local learning rates as the training progresses. In addition, FEDHYPER not only showcases unparalleled robustness to a spectrum of initial learning rate configurations but also significantly alleviates the necessity for laborious empirical learning rate adjustments. We provide a comprehensive theoretical analysis of FEDHYPER's convergence rate and conduct extensive experiments on vision and language benchmark datasets. The results demonstrate that FEDHYPER consistently converges 1.1-3$\times$ faster than FEDAVG and the competing baselines while achieving superior final accuracy. Moreover, FEDHYPER catalyzes a remarkable surge in accuracy, augmenting it by up to 15% compared to FEDAVG under suboptimal initial learning rate settings. Our source code is released at https://github.com/ATP-1010/FedHyper.

1 INTRODUCTION

Federated Learning (FL) (Zhang et al., 2021; McMahan et al., 2017; Reddi et al., 2020) has emerged as a popular distributed machine learning paradigm in which a server orchestrates the collaborative training of a machine learning model across massive distributed devices. The clients only need to communicate local model updates with the server without explicitly sharing the private data. FL has been widely applied in numerous applications, such as the next-word prediction in a virtual keyboard on the smartphone (Hard et al., 2018) and hospitalization prediction for patients (Brisimi et al., 2018). However, FL presents challenges compared to conventional distributed training, including communication bottlenecks, data heterogeneity, privacy concerns, etc (Kairouz et al., 2021).

Difficulty of Scheduling Learning Rates in FL. In each communication round of FL, the selected clients usually perform several epochs of local stochastic gradient descent (SGD) to optimize their local models before communicating with the server in FEDAVG. The server then updates the global model by the aggregated client updates Li et al. (2019); Karimireddy et al. (2020). According to this process, FL involves two types of learning rates, i.e., *global learning rate* on the server to update the global model and *local learning rate* for optimizing the local objectives on clients. They both significantly impact the model accuracy and convergence speed (Jhunjhunwala et al., 2023), while their optimal values can be influenced by various factors such as the dataset, model structure, and training stage (Reddi et al., 2020). Suboptimal learning rates can hinder model convergence, i.e., small learning rates slow down convergence while aggressive learning rates prohibit convergence (Barzilai & Borwein, 1988). Therefore, scheduling the learning rates is crucial for both speeding up convergence and improving the model performance in FL, as observed previously in centralized machine learning (You et al., 2019), Nevertheless, one of the critical challenges in FL is the data heterogene-

ity across clients Rush et al. (2023), the local optimization objectives might significantly diverge from the global one. Therefore, the server and clients need to cooperatively schedule learning rates to foster their synergy. This makes the scheduler in FL very complicated. Merely applying the optimizers in traditional machine learning, such as Adam (Kingma & Ba, 2014) and Adagrad (Lydia & Francis, 2019), to FL cannot consistently work well across different FL tasks.

In addition to the aforementioned complexities, configuring appropriate initial learning rates is also challenging for FL due to heterogeneous optimization objectives across clients. It is impractical to explore different initial learning rates due to expensive communication and computation costs. Therefore, it is necessary to design a scheduler that is robust against random initial learning rates.

Previous Works. Recent studies in speeding up FL have focused on adjusting the learning rate through optimizers or schedulers, primarily emphasizing the *global* model update. For instance, weight decay (Yan et al., 2022) is used to decrease the global learning rate as training progresses but exhibits much slower convergence when its initial learning rate is below the optimal value. Alternative methods, such as (Reddi et al., 2020), harness popular optimizers in centralized machine learning, i.e., Adam and Adagrad, for global updates. However, a very recent work, FEDEXP (Jhunjhunwala et al., 2023), reveals that these methods are not efficient enough in FL. FEDEXP adjusts the global learning rate based on local updates and outperforms the methods aforementioned. However, it exhibits inconsistent performance across tasks and can result in unstable accuracy in the late training stages. More importantly, these approaches mainly consider the server's perspective on global learning rate adjustments. We argue for a scheduler that integrates both global and local rates, enhancing server-client collaboration. Additionally, current solutions generally fail to ensure robustness against variations in the initial learning rate, with the exception of FEDEXP, which only maintains resilience against the initial global learning rate.

Therefore, we seek to answer a compelling research question: *How can we schedule both global and local learning rates according to the training progress and the heterogeneous local optimization objectives while ensuring robustness against the random initial learning rate?*

Our Solution. To tackle the question above, we design FEDHYPER, a universal and robust learning rate scheduler for FL. Specifically, FEDHYPER functions as a versatile toolbox, facilitating the scheduling of learning rates both on the server and among the clients. It encompasses (1) a *global scheduler* that schedules the global learning rate on the server; (2) a *server-side local scheduler* that tunes the local learning rate on the server; (3) a *client-side local scheduler* that adjusts the local learning rate between local epochs on the clients. These schedulers can be either independently or cooperatively applied to FL tasks, aligning with user specifications. Moreover, FEDHYPER is also seamlessly compatible with current FL optimization algorithms, thereby further improving their efficacy, such as server momentum (Liu et al., 2020) and FEDADAM (Reddi et al., 2020).

The key idea of FEDHYPER, inspired by (Chen et al., 2022; Baydin et al., 2017), is to utilize the *hypergradient* (Maclaurin et al., 2015) of the learning rate to guide the scheduling. We begin by recalling the theoretical analysis of the relationship between the optimization objective and the learning rates. The analysis reveals that the hypergradient of the learning rate is influenced by the model gradients from the current and previous training steps. Based on this insight, we formally define the hypergradients of local and global learning rates in FL by exploiting the current and historical model updates. Consequently, we design FEDHYPER, consisting of the three aforementioned schedulers. As the hypergradient of the learning rate is based on the model updates, it can precisely capture the dynamics of the training progress. Therefore, FEDHYPER can effectively schedule random initial learning rates to optimal values, making it robust against different initial learning rates. Through experiments spanning computer vision and natural language tasks, we demonstrate that FEDHYPER effectively fosters convergence, facilitates superior accuracy, and enhances the robustness against random initial learning rates. Our contributions are summarized as follows:

- We conduct a meticulous theoretical analysis for the hypergradients of local and global learning rates with regard to the gradients of local and global models.
- Based on the analysis described above, we designed FEDHYPER, a universal and robust learning rate scheduling algorithm, which can adjust both global and local learning rates and enhance the robustness against random initial learning rates.
- We propose a novel theoretical framework of FL by considering the dynamic local and global learning rates, showing that FEDHYPER is guaranteed to converge in FL scenarios.

- We conduct extensive experiments on vision and language tasks. The results demonstrate not only the efficacy of FEDHYPER to improve both convergence speed and accuracy, but also its robustness under various initial learning rate settings. FEDHYPER convergences 1.1-3x faster than FEDAVG and most competing baselines, and increases accuracy by up to 15% in various initial learning rate settings. Moreover, plugging FEDHYPER into current FL optimization methods also exhibits additional performance improvement.

2 RELATED WORK

Hypergradient Descent of Learning Rates. Hypergradient descent is commonly used to optimize the learning rate due to its integral role in the gradient descent process. Baydin et al. introduced HD (Baydin et al., 2017), an algorithm that approximates the current learning rate's hypergradient using the gradients from the last two epochs. Building on this, the Differentiable Self-Adaptive (DSA) learning rate (Chen et al., 2022) was proposed, which precomputes the next epoch's gradient and uses it, along with the current epoch's gradient, to determine the hypergradient. While DSA offers enhanced accuracy over HD, it requires greater computational resources. Moreover, studies like (Jie et al., 2022) have explored optimizing the meta-learning rate in hypergradient descent to further improve HD's efficiency.

Hyperparameter Optimization in FL. Hyperparameter optimization is crucial in FL. While (Wang et al., 2022) provides a benchmark, other studies like Flora (Zhou et al., 2021) and (Zhou et al., 2022) focus on hyperparameter initialization using single-shot methods. (Agrawal et al., 2021) clusters clients by data distribution to adjust hyperparameters, and FedEx (Khodak et al., 2020) employs weight sharing for streamlined optimization. Some research, such as (Kan, 2022), targets specific hyperparameters, while (Shi et al., 2022) dynamically changes the batch size during FL training. Despite these efforts, a comprehensive tool for hyperparameter optimization in FL is lacking. Our work addresses this gap, offering enhanced versatility and practicality.

3 PROPOSED METHOD: FEDHYPER

Preliminaries on Hypergradient. In general machine learning, we aim at minimizing an empirical risk function $F(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{w}; s_i)$, where $\boldsymbol{w} \in \mathbb{R}^d$ denotes the model weight and s_i denotes the i-th sample in the training dataset. The most common way to minimize this loss function is to use gradient-based methods (Bottou, 2010), the update rule of which is given as follows:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta^{(t)} \nabla F(\boldsymbol{w}^{(t)}), \tag{1}$$

where $\eta^{(t)}$ is a time-varying scalar learning rate. The choice of η significantly influences the training procedure. A greedy approach to set η is to select the value that can minimize the updated loss (i.e., $\min_\eta F(\boldsymbol{w}^{(t)} - \eta \nabla F(\boldsymbol{w}^{(\boldsymbol{t})}))$). However, this is infeasible due to the need to evaluate countless possible values. Hypergradient follows this idea and allows the learning rate to be updated by a gradient as well. Specifically, when taking the derivation, one can get

$$\frac{\partial F(\boldsymbol{w}^{(t+1)})}{\partial \eta^{(t)}} = \nabla F(\boldsymbol{w}^{(t+1)}) \cdot \frac{\partial(\boldsymbol{w}^{(t)} - \eta^{(t)} \nabla F(\boldsymbol{w}^{(t)}))}{\partial \eta^{(t)}} \tag{2}$$

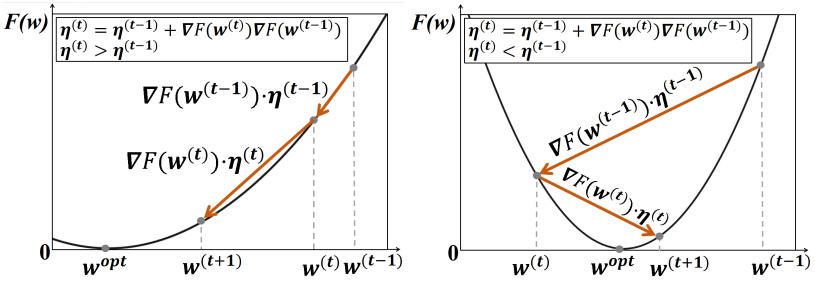
$$= -\nabla F(\boldsymbol{w}^{(t+1)}) \cdot \nabla F(\boldsymbol{w}^{(t)}). \tag{3}$$

Ideally, in order to get $\eta^{(t)}$, the current learning rate $\eta^{(t-1)}$ should be updated by $-\nabla F(\boldsymbol{w}^{(t+1)}) \cdot \nabla F(\boldsymbol{w}^{(t)})$. But this requires future knowledge $\nabla F(\boldsymbol{w}^{(t+1)})$. By assuming the optimal values of η do not change much between consecutive iterations, $\eta^{(t-1)}$ is updated as follows:

$$\eta^{(t)} = \eta^{(t-1)} - \Delta_\eta^{(t)} \tag{4}$$

$$= \eta^{(t-1)} + \nabla F(\boldsymbol{w}^{(t)}) \cdot \nabla F(\boldsymbol{w}^{(t-1)}), \tag{5}$$

where $\Delta_\eta^{(t)} = -\nabla F(\boldsymbol{w}^{(t)}) \cdot \nabla F(\boldsymbol{w}^{(t-1)})$ is defined as the hypergradient. This learning rate update rule is very intuitive, allowing adaptive updates based on the training dynamics. If the inner product $\nabla F(\boldsymbol{w}^{(t)}) \cdot \nabla F(\boldsymbol{w}^{(t-1)})$ is positive, i.e., the last two steps' gradients have similar directions, then the learning rate will increase to further speedup the good progress (see Figure 1(a)). If the inner product is negative, i.e., two consecutive gradients point to drastically diverged directions, then the learning rate will decrease to mitigate oscillations (see Figure 1(b)).



(a) $\nabla F(\boldsymbol{w}^{(t)}) \cdot \nabla F(\boldsymbol{w}^{(t-1)}) > 0$ (b) $\nabla F(\boldsymbol{w}^{(t)}) \cdot \nabla F(\boldsymbol{w}^{(t-1)}) < 0$

Figure 1: Two situations in hypergradient descent. Hypergradient descent adjusts the learning rate according to the inner product of gradients in the last two iterations.

Applying Hypergradient to FL. In FL, the goal is to optimize a global objective function $F(\boldsymbol{w})$ which is defined as a weighted average over a large set of local objectives:

$$F(\boldsymbol{w}) = \frac{1}{M} \sum_{m=1}^{M} F_m(\boldsymbol{w}), \tag{6}$$

where M is the number of clients and $F_m(\boldsymbol{w}) = \frac{1}{N_m} \sum_n f_m(\boldsymbol{w}; s_n)$ is the local loss function of the m-th client. Due to privacy concerns, the server cannot access any client data and clients can neither share data with others. FEDAVG (McMahan et al., 2017) is the most popular algorithm to solve this problem. It allows clients to perform local training on their own local dataset. A central coordinating server only aggregates the model weight changes to update the server model:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha^{(t)} \frac{1}{M} \sum_{m=1}^{M} \Delta_m^{(t)}, \tag{7}$$

where $\alpha^{(t)}$ is defined as the server learning rate, and $\Delta_m^{(t)}$ is the local model changes. Moreover, the local update rule on each client is:

$$\boldsymbol{w}_m^{(t,k+1)} = \boldsymbol{w}_m^{(t,k)} - \beta^{(t,k)} g_m(\boldsymbol{w}_m^{(t,k)}), \tag{8}$$

$$\Delta_m^{(t)} = \boldsymbol{w}_m^{(t,0)} - \boldsymbol{w}_m^{(t,\tau)} = \boldsymbol{w}^{(t)} - \boldsymbol{w}_m^{(t,\tau)}, \tag{9}$$

where $\beta^{(t,k)}$ is the local learning rate, $\boldsymbol{w}_m^{(t,k)}$ denote the local model at m-th client at the t-th global round and k-th local step, and $\boldsymbol{w}_m^{(t,0)} = \boldsymbol{w}^{(t)}$ is the initial model at each round.

Comparing the update rules of FEDAVG with vanilla gradient descent, the learning rate scheduling for FedAvg can be way more complicated. When applying the vanilla hypergradient idea on the server learning rate α, one will find that the global gradient $\nabla F(\boldsymbol{w})$ is not available at all; when applying the idea on the client learning rate, the derivative with respect to $\beta^{(t,k)}$ is nearly intractable.

In order to address these challenges, we develop FEDHYPER, a learning rate scheduler based on hypergradient descent specifically designed for FL. FEDHYPER comprises a global scheduler FEDHYPER-G, a server-side local scheduler FEDHYPER-SL, and a client-side local scheduler FEDHYPER-CL. The general learning rates update rules are given below:

$$\text{FEDHYPER-G: } \alpha^{(t)} = \alpha^{(t-1)} - \Delta_\alpha^{(t-1)}, \tag{10}$$

$$\text{FEDHYPER-SL: } \beta^{(t,0)} = \beta^{(t-1,0)} - \Delta_{\beta,s}^{(t-1)}, \tag{11}$$

$$\text{FEDHYPER-CL: } \beta^{(t,k)} = \beta^{(t,k-1)} - \Delta_{\beta,c}^{(t,k-1)}. \tag{12}$$

In the following subsections, we will specify the exact expressions of each hypergradient.

3.1 FEDHYPER-G: USING HYPERGRADIENT FOR THE GLOBAL LEARNING RATE

We first provide FEDHYPER-G to adjust the global learning rate α based on the hypergradient descent. According to Eq. (1), the global learning rate for the tth round $\alpha^{(t)}$ should be computed by

the last global learning rate $\alpha^{(t-1)}$ and global gradients in the two preceding rounds. Nevertheless, in accordance with the global model updating strategy in Eq. (7), the global model is updated by the aggregation of local model updates rather than gradients. Therefore, in FEDHYPER-G, we treat the aggregation of the local model updates as the pseudo global gradient $\Delta^{(t)}$:

$$\Delta^{(t)} = \frac{1}{M} \sum_{m=1}^{M} \Delta_m^{(t)} \tag{13}$$

Then, we replace the gradients in Eq. (5) with global model updates and obtain the $\Delta_\alpha^{(t-1)}$ in Eq. (10) for the global scheduler as follows:

$$\Delta_\alpha^{(t-1)} = -\Delta^{(t)} \cdot \Delta^{(t-1)}, \tag{14}$$

$$\alpha^{(t)} = \alpha^{(t-1)} + \Delta^{(t)} \cdot \Delta^{(t-1)} \tag{15}$$

In addition, to ensure convergence and prevent gradient exploration, we clip $\alpha^{(t)}$ by a preset bound value γ_α in each round:

$$\alpha^{(t)} = \min\{\max\{\alpha^{(t)}, \frac{1}{\gamma_\alpha}\}, \gamma_\alpha\} \tag{16}$$

Through experiments, we have found that $\gamma_\alpha = 3$ yields consistent and stable performance across various tasks and configurations. Therefore, we generally do not need to further adjust γ_α.

3.2 FEDHYPER-SL & FEDHYPER-CL: USING HYPERGRADIENT FOR THE CLIENT LEARNING RATE

FedHyper-SL. The server-side local scheduler adopts the same hypergradient as the global scheduler, i.e., $\Delta_{\beta,s}^{(t-1)} = \Delta_\alpha^{(t-1)}$. Thus, the local learning rates are updated by the server as follows:

$$\beta^{(t,0)} = \beta^{(t-1,0)} + \Delta^{(t)} \cdot \Delta^{(t-1)}. \tag{17}$$

It adjusts all the selected local learning rates from the server synchronously. These local learning rates are also clipped by a preset bound γ_β:

$$\beta^{(t,0)} = \min\{\max\{\beta^{(t,0)}, \frac{1}{\gamma_\beta}\}, \gamma_\beta\}. \tag{18}$$

The updated local learning rates $\beta^{(t,0)}$ are sent to clients *after* t-th round and are used by clients in the $t+1$-th round. Like FEDHYPER-G, we have experimentally set γ_β to 10. This bound will also be used for $\beta^{(t,k)}$ in FEDHYPER-CL.

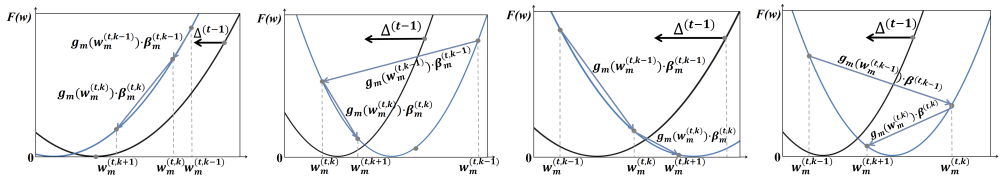
FedHyper-CL. Adjusting the local learning rate from the clients is a more fine-grained scheduling strategy since it adjusts learning rates for each local batch, while the server only performs such adjustments between communication rounds. This client-centric approach not only enhances convergence speed but also outperforms both FEDHYPER-G and FEDHYPER-SL in terms of accuracy. According to the hypergradient descent in Eq. (5) and local model update rule Eq. (8), the local learning rate for the k-th epoch on client m can be updated as follows:

$$\Delta_{\beta,c}^{(t,k-1)} = -g_m(\boldsymbol{w}_m^{(t,k)}) \cdot g_m(\boldsymbol{w}_m^{(t,k-1)}). \tag{19}$$

Nevertheless, directly applying Eq. (19) to local learning rates can lead to an imbalance in learning rates across clients, which may lead to convergence difficulty. To address this issue, and draw inspiration from our global scheduler, we incorporate global model updates to regulate the local learning rate optimization. Specifically, we augment $\Delta_{\beta,c}^{(t,k-1)}$ by adding the component $\frac{1}{K} \cdot g_m(\boldsymbol{w}_m^{t,k}) \cdot \Delta^{t-1}$, where Δ^{t-1} is the global model update in the last round and K is the number of local SGD steps:

$$\beta^{(t,k)} = \beta^{(t,k-1)} + g_m(\boldsymbol{w}_m^{(t,k)}) \cdot g_m(w_m^{(t,k-1)}) + \frac{1}{K} \cdot g_m(\boldsymbol{w}_m^{(t,k)}) \cdot \Delta^{(t-1)}. \tag{20}$$

The term $g_m(\boldsymbol{w}_m^{(t,k)}) \cdot \Delta^{(t-1)}$ will be positive if $g_m(\boldsymbol{w}_m^{(t,k)})$ and Δ^{t-1} have similar direction, otherwise it is negative. The coefficient $\frac{1}{K}$ acts to normalize the magnitude of $g_m(\boldsymbol{w}_m^{(t,k)}) \cdot \Delta^{(t-1)}$ such



(a) $X > 0; Y > 0$ (b) $X < 0; Y < 0$ (c) $X > 0; Y < 0$ (d) $X < 0; Y > 0$

Figure 2: Four situations in client-side local learning rate scheduling.

that it aligns with the scale of $g_m(\boldsymbol{w}_m^{(t,k)}) \cdot g_m(\boldsymbol{w}_m^{(t,k-1)})$. This normalization is essential given that while $g_m(\boldsymbol{w}_m^{(t,k)})$ is the gradient of a single SGD step, Δ^{t-1} is the average over K local SGD steps. By integrating this term, we increase $\beta^{(t,k)}$ when the local model update is directionally consistent with the global model update; conversely, we decrease it when they exhibit significant discrepancies. It is pivotal to prevent certain clients from adopting excessively high learning rates that could potentially impact convergence.

Here we define $X = g_m(\boldsymbol{w}_m^{(t,k)}) \cdot g_m(\boldsymbol{w}_m^{(t,k-1)}); Y = \frac{1}{K} \cdot g_m(\boldsymbol{w}_m^{(t,k)}) \cdot \Delta^{(t-1)}$. The local learning rate in the following four situations will be adjusted as: (1) $X > 0; Y > 0$ as in Figure 2(a), $\beta^{(t,k)}$ will increase; (2) $X < 0; Y < 0$ as in Figure 2(b), $\beta^{(t,k)}$ will decrease; (3) $X > 0; Y < 0$ as in Figure 2(c), $\beta^{(t,k)}$ will increase if $|X| > |Y|$; otherwise, it will decrease; (4) $X < 0; Y > 0$ as in Figure 2(d), $\beta^{(t,k)}$ will increase if $|X| < |Y|$; otherwise, it will undergo a decrease.

The client-side local scheduler integrates insights from both global and local training dynamics. By leveraging global model updates, it steers the local learning rate optimization effectively. The full algorithm of FEDHYPER is obtained in Appendix C.

4 CONVERGENCE GUARANTEE

In this section, we analyze the convergence of FEDHYPER considering dynamic global and local learning rates. Our analysis demonstrate that FEDHYPER can ensure convergence given a sufficient number of training epochs under some standard assumptions (Wang et al., 2020b).

According to the unified convergence analysis in FEDNOVA (Wang et al., 2020b), the convergence of an FL framework based on the objective in Eq. (6) can be proved with some basic assumptions, which we list in Appendix A.

In addition, we have the following upper and lower bounds for global and local learning rates:

Constraint 1 The learning rates satisfy $\frac{1}{\gamma_\alpha} \leq \alpha^{(t)} \leq \gamma_\alpha$ and $\frac{1}{\gamma_\beta} \leq \beta^{(t,k)} \leq \gamma_\beta$, where $\gamma_\alpha, \gamma_\beta > 1$.

Theorem 1 Under Assumption 1 to 3 in Appendix A and constraint 1 above, we deduce the optimization error bound of FEDHYPER from the convergence theorem of FEDNOVA.

$$\min_{t \in [T]} \mathbb{E}\|\nabla F(\boldsymbol{w}^{(t)})\|^2 < O\left(\frac{P}{\sqrt{MkT}} + \frac{Q}{kT}\right), \tag{21}$$

where quantities P, Q are defined as follows:

$$P = (\sum_{m=1}^{M} \frac{\gamma_\beta^4 \sigma^2}{M} + 1)\gamma_\alpha, \tag{22}$$

$$Q = \sum_{m=1}^{M} \left[[(k-1)\gamma_\beta^4 + 1]^2 - \frac{1}{\gamma_\beta^2}\right] \sigma^2 + M\rho^2 \left[[(k-1)\gamma_\beta^2 + 1]^2 - \frac{1}{\gamma_\beta^2}[\frac{k-1}{\gamma_\beta^2} + 1]\right]. \tag{23}$$

Where σ and ρ are constants that satisfy $\sigma^2, \rho^2 \geq 0$. The bound defined by Eq. (21), (22), and (23) indicates that when the number of rounds T tends towards infinity, the expectation of $\nabla F(\boldsymbol{w}^{(t)})$ tends towards 0. We further provide a comprehensive proof of Theorem 1 in Appendix A.

5 EXPERIMENTS

We evaluate the performance of FEDHYPER on three benchmark datasets, including FMNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., 2009) for image classification, and Shakespeare (McMahan et al., 2017) for the next-word prediction task. To evaluate the efficacy of adapting the global learning rate, we compare FEDHYPER-G against four baselines: FEDADAGRAD (Reddi et al., 2020), FEDADAM (Reddi et al., 2020), FEDEXP (Jhunjhunwala et al., 2023) and global learning rate decay with a 0.995 decay factor. We employ the across-round local learning rate step decay with also a 0.995 decay factor as the baseline to compare with the server-side local scheduler FEDHYPER-SL. As for client-side local scheduler FEDHYPER-CL, we opt for FEDAVG paired with local SGD and local Adam as the comparative baselines.

Experimental Setup. We partition FMNIST and CIFAR10 dataset into 100 clients following a Dirichlet distribution with $\alpha = 0.5$ as presented in(Wang et al., 2020a). We directly partition the inherently non-iid Shakespeare dataset to 100 clients as well. In each communication round, the server randomly selects 10 clients to perform local SGD and update the global model. For FMNIST, we utilize a CNN model, while for CIFAR10, we deploy a ResNet-18 (He et al., 2016). For the next-word prediction task on Shakespeare, we implement a bidirectional LSTM model (Graves et al., 2005). We set our global bound parameter γ_α at 3 and the local bound γ_β at 10 in all experiments. In order to achieve global model convergence, we conduct 50, 200, and 600 communication rounds for FMNIST, CIFAR10, and Shakespeare, respectively.

5.1 FEDHYPER COMPREHENSIVELY OUTPERFORMS FEDAVG AND BASELINES

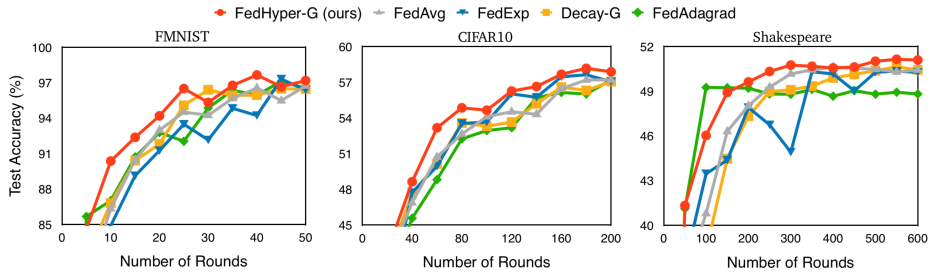
Performance of FedHyper-G. As Figure 3(a) illustrates, the results underscore the superiority of FEDHYPER-G over the current global optimizers and schedulers in FL. Across all three datasets, FEDHYPER-G achieves faster convergence rates in the early training stage (e.g., rounds 0 to 25 in FMNIST, rounds 0 to 80 in CIFAR10, and rounds 0 to 300 in Shakespeare) than all baselines. Moreover, FEDHYPER-G also maintains an equal or superior accuracy than baselines.

Such results corroborate our intuition in Section 3: FEDHYPER can expedite the training process when the model remains far from the optimum and enhance the accuracy. Additionally, FEDHYPER-G increases accuracy by 0.45% 0.58%, and 0.66% compared to the best baseline in FMNIST, CIFAR10, and Shakespeare, respectively. This underscores the consistent performance improvement of FEDHYPER-G across varied tasks. Although FEDEXP shows comparable performance as FEDHYPER-G in FMNIST and CIFAR10 for image classification, FEDHYPER-G outperforms FEDEXP on Shakespeare dataset.

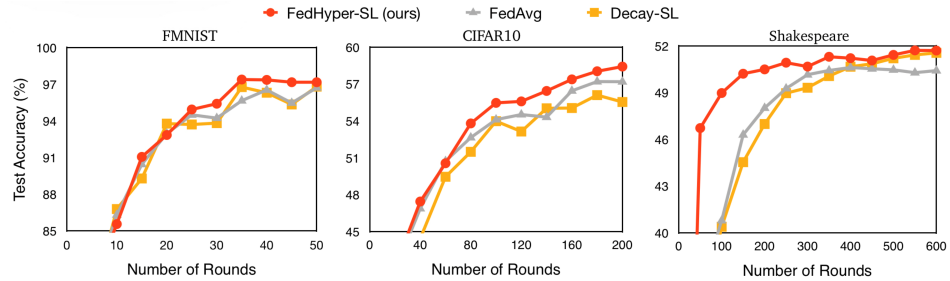
Performance of FedHyper-SL. We compare FEDHYPER-SL with FEDAVG and across-round local learning rate decay, i.e., Decay-SL. As Figure 3(b) shows, FEDHYPER-SL can potentially lead to comparable, if not superior, convergence speed and accuracy when contrasted with FEDHYPER-G. This observation is particularly pronounced in the CIFAR10 and Shakespeare datasets. For instance, in the Shakespeare dataset, FEDHYPER-SL achieves a 50% test accuracy approximately by the 150th round, whereas FEDHYPER-G reaches a similar accuracy near the 250th round. A plausible explanation for this might be that the local learning rate impacts every local SGD iteration, in contrast to the global scheduler, which only impacts the aggregation process once per round.

Performance of FedHyper-CL. Figure 3(c) exhibits the efficacy of the client-side local scheduler. Not only does FEDHYPER-CL outperform the baselines, but it also manifests superior performance compared to our other two schedulers, as analyzed in Section 3. Specifically, in FMNIST and CIFAR10, it outperforms FEDAVG with SGD and Adam, with improvements of 0.44% and 1.18%, respectively. Regarding the Shakespeare dataset, FEDHYPER-CL excels over FEDAVG with local SGD by 1.33%, though its performance aligns closely with that of FEDAVG with local Adam. Overall, FEDHYPER-CL consistently demonstrates commendable results across diverse tasks.

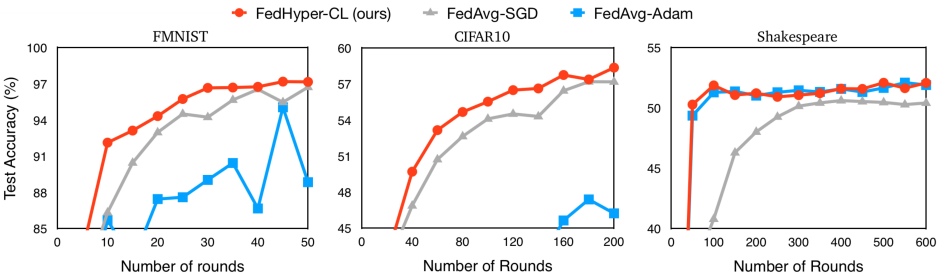
Additionally, FEDHYPER-CL also shows a better performance compared to FEDHYPER-G and FEDHYPER-SL, most notably in terms of convergence speed. For example, FEDHYPER-CL achieves 52.08% accuracy, converging approximately by the 100th round. In contrast, FEDHYPER-G and FEDHYPER-SL yield accuracies of 51.08% and 51.70% respectively, with a protracted convergence around the 300th round. It is noteworthy that FEDHYPER-CL does incur a higher computa-



(a) Performance of FedHyper-G



(b) Performance of FedHyper-SL



(c) Performance of FedHyper-CL

Figure 3: Comparison of FedHyper with baselines. FEDHYPER consistently gives faster convergence compared to baselines with superior performance.

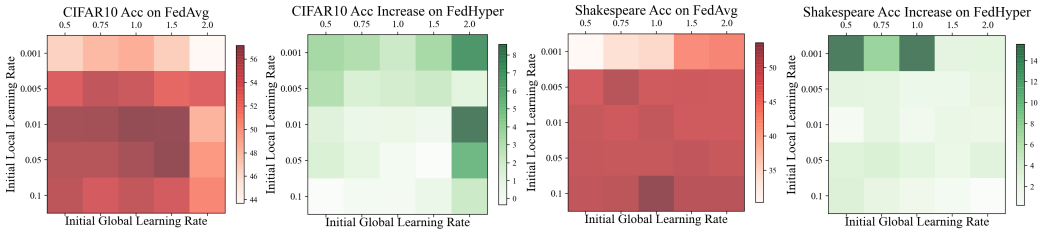
tional overhead compared to the other two methods in FEDHYPER. Consequently, a comprehensive discussion regarding the selection of three schedulers will be presented in the Appendix B.

5.2 DISCUSSION AND ABLATIONS OF FEDHYPER

Robustness of FEDHYPER Against Initial Learning Rates. As discussed in Section 1, identifying the optimal initial learning rates in FL presents a significant challenge. FEDHYPER facilitates faster scheduling of the learning rates, showing its versatility to attain desired accuracy even in scenarios characterized by suboptimal initial learning rates, especially when contrasted with FEDAVG.

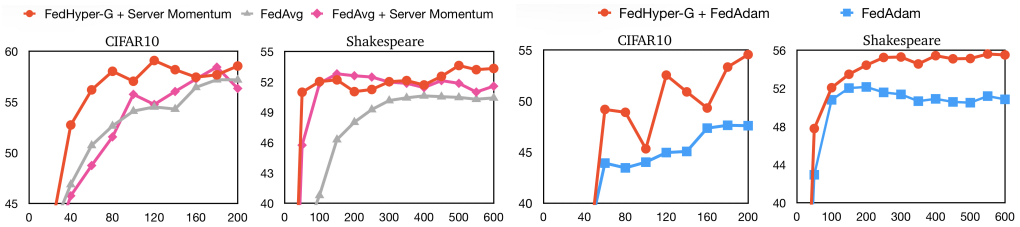
We first conduct experiments on CIFAR10 and Shakespeare to evaluate the robustness of FEDHYPER against varied initial learning rates. Specifically, we employ five distinct initial global learning rates: 0.5, 0.75, 1, 1.5, and 2. Additionally, we also consider five separate initial local learning rates: 0.001, 0.005, 0.01, 0.05, and 0.1. We first train CIFAR10 and Shakespeare using FEDAVG by applying the aforementioned learning rates. The final accuracy is depicted in Figure 4(a) and Figure 4(c). The darker red shades represent higher final test accuracy.

Following the initial experiments, we apply FEDHYPER in conjunction with both the global scheduler and the client-side local scheduler, denoted as FEDHYPER-G + FEDHYPER-CL, employing the identical initial learning rates as mentioned previously. The results are illustrated in Figure 4(b) and Figure 4(d), the shades of green serve as indicators of the magnitude of accuracy augmentation, with the darker green hues representing more substantial improvements in accuracy.



(a) (b) (c) (d)

Figure 4: FEDHYPER's robustness against diverse initial learning rates.



(a) FEDHYPER + server momentum (b) FEDHYPER + FEDADAM

Figure 5: Integration of FEDHYPER with server momentum and FEDADAM.

The results in Figure 4 show that FEDHYPER improves accuracy by 1% - 5% over FEDAVG for most initial learning rate configurations. Notably, FEDHYPER is particularly effective at enhancing the accuracy of FEDAVG when it is faced with suboptimal initial learning rates. For instance, in the Shakespeare dataset (see Figure 4(c)), FEDAVG excels with higher initial local learning rates (> 0.005), but underperforms at suboptimal initial local learning rates like 0.001. In such scenarios, Figure 4(d) demonstrates FEDHYPER's ability to substantially boost accuracy, notably in cases with a 0.001 initial local learning rate. When paired with a 0.5 initial global rate, FEDAVG achieves 30% accuracy, whereas FEDHYPER raises it by almost 15% to 45.77%. The robustness of FEDHYPER against random initial learning rates can be visually demonstrated in Figure 4, i.e., areas characterized by a lighter shade of red (i.e., indication of lower accuracy) in Figure 4(a) and Figure 4(c) transform into darker shades of green (representation of significant accuracy augmentation) in Figure 4(b) and Figure 4(d). The robustness manifested by FEDHYPER thus mitigates the necessity for meticulous explorations to initial learning rates within the FL paradigm.

Integrating FedHyper with Existing Optimizers. We evaluate the efficacy of FEDHYPER-G when it is integrated with existing global optimization algorithms, i.e., server momentum (Liu et al., 2020) and FEDADAM (Reddi et al., 2020). The results in Figure 5 underscore that FEDHYPER is capable of improving the performance of these widely-adopted FL optimizers. For instance, FEDHYPER-G together with server momentum shows superior performance over the combination of FEDAVG and server momentum. Moreover, as Figure 5(b) depicts, when FEDHYPER-G is paired with FEDADAM for the CIFAR10, there is a marked enhancement in performance. These findings demonstrate that FEDHYPER possesses the versatility to serve not merely as a universal scheduler, but also as a universally applicable enhancement plugin that can elevate the capabilities of existing optimizers.

6 CONCLUSION

In this paper, we introduced FEDHYPER, a universal and robust learning rate scheduling algorithm rooted in hypergradient descent. FEDHYPER can optimize both global and local learning rates throughout the training. Our experimental results have shown that FEDHYPER consistently outperforms baseline algorithms in terms of convergence rate and test accuracy. Our ablation studies demonstrate the robustness of FEDHYPER under varied configurations of initial learning rates. Furthermore, our empirical evaluations reveal that FEDHYPER can seamlessly integrate with and augment the performance of existing optimization algorithms.

REFERENCES

Shaashwat Agrawal, Sagnik Sarkar, Mamoun Alazab, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Quoc-Viet Pham, et al. Genetic cfl: Hyperparameter optimization in clustered federated learning. *Computational Intelligence and Neuroscience*, 2021, 2021.

Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pp. 177–186. Springer, 2010.

Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

Bozhou Chen, Hongzhi Wang, and Chenmin Ba. Differentiable self-adaptive learning rate. *arXiv preprint arXiv:2210.10290*, 2022.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pp. 799–804. Springer, 2005.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedexp: Speeding up federated averaging via extrapolation. *arXiv preprint arXiv:2301.09604*, 2023.

Renlong Jie, Junbin Gao, Andrey Vasnev, and Minh-Ngoc Tran. Adaptive hierarchical hypergradient descent. *International Journal of Machine Learning and Cybernetics*, 13(12):3785–3805, 2022.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Andrew K Kan. Federated hypergradient descent. *arXiv preprint arXiv:2211.02106*, 2022.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8):1754–1766, 2020.

Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

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

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Keith Rush, Zachary Charles, and Zachary Garrett. Federated automatic differentiation. *arXiv preprint arXiv:2301.07806*, 2023.

Dian Shi, Liang Li, Maoqiang Wu, Minglei Shu, Rong Yu, Miao Pan, and Zhu Han. To talk or to work: Dynamic batch sizes assisted time efficient federated learning over future mobile edge devices. *IEEE Transactions on Wireless Communications*, 21(12):11038–11050, 2022.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020a.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Zhen Wang, Weirui Kuang, Ce Zhang, Bolin Ding, and Yaliang Li. Fedhpo-b: A benchmark suite for federated hyperparameter optimization. *arXiv preprint arXiv:2206.03966*, 2022.

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

Gang Yan, Hao Wang, and Jian Li. Seizing critical learning periods in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8788–8796, 2022.

Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.

Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

Yi Zhou, Parikshit Ram, Theodoros Salonidis, Nathalie Baracaldo, Horst Samulowitz, and Heiko Ludwig. Flora: Single-shot hyper-parameter optimization for federated learning. *arXiv preprint arXiv:2112.08524*, 2021.

Yi Zhou, Parikshit Ram, Theodoros Salonidis, Nathalie Baracaldo, Horst Samulowitz, and Heiko Ludwig. Single-shot hyper-parameter optimization for federated learning: A general algorithm & analysis. *arXiv preprint arXiv:2202.08338*, 2022.