A PROOF OF THEOREM 1: CONVERGENCE OF FEDHYPER

According to theoretical analysis of FEDNOVA (Wang et al., 2020b), an FL framework that follows the update rule in Eq. (13) will converge to a stationary point, and the optimization error will be bounded with the following assumptions:

Assumption 1 (Smoothness). Each local objective function is Lipschitz smooth, that is, $\|\nabla F_m(x) - \nabla F_m(y)\| \leq L\|x-y\|, \forall m \in \{1,2,...,M\}$.

Assumption 2 (Unbiased Gradient and Bounded Variance). The stochastic gradient at each client is an unbiased estimator of the local gradient: $E_\xi[g_i(x|\xi)] = \nabla F_m(x)$ and has bounded variance $E_\xi[\|g_m(x|\xi) - \nabla F_m(x)\|^2] \leq \sigma^2, \forall m \in \{1,2,...,M\}, \sigma^2 \geq 0$.

Assumption 3 (Bounded Dissimilarity). Existing constants $\psi^2 \geq 1, \rho^2 \geq 0$ such that $\Sigma_{m=1}^M \frac{1}{M}\|\nabla F_m(x)\|^2 \leq \psi^2 \|\Sigma_{m=1}^M \frac{1}{M}\nabla F_m(x)\|^2$. If local functions are identical to each other, then we have $\psi^2 = 1, \rho^2 = 0$.

where we adhere to our setting that each client contributes to the global model with the equal weight $\frac{1}{M}$. Then we can rewrite the optimization error bound as follows:

$$\min_{t\in[T]} \mathbb{E}\|\nabla F(w^{(t)})\|^2 \leq O\left(\frac{\alpha^{(t)}}{\sqrt{MkT}}\right) + O\left(\frac{A\sigma^2}{\sqrt{MkT}}\right) + O\left(\frac{MB\sigma^2}{kT}\right) + O\left(\frac{MC\rho^2}{kT}\right), \quad (24)$$

Where A, B, and C are defined by:

$$A = \alpha^{(t)} \sum_{m=1}^{M} \frac{\|a_m\|_2^2}{M\|a_m\|_1^2}, B = \sum_{m=1}^{M} \frac{1}{M}(\|a_m\|_2^2 - a_{m,-1}^2), C = \max_m\{\|a_m\|_1^2 - \|a_m\|_1 a_{m,-1}\}. \quad (25)$$

where a_m is a vector that can measure the local model update during local SGD, where the number of k-th value of a_m is $a_m[k] = \frac{\beta^{(t,k)}}{\beta^{(t,0)}}$.

Note that we have Bound 1 on global learning rate that $\alpha^{(t)} = \min\{\max\{\alpha^t, \frac{1}{\gamma_\alpha}\}, \gamma_\alpha\}$, so we have the upper and lower bound for $\alpha^{(t)}$ as follows:

$$\frac{1}{\gamma_\alpha} \leq \alpha^{(t)} \leq \gamma_\alpha, \quad (26)$$

For the local learning rate, we have $\beta^{(t,k)} = \min\{\max\{\beta^{(t,k)}, \frac{1}{\gamma_\beta}\}, \gamma_\beta\}$. Therefore, the maximum value of ratio $\frac{\beta^{(t,k)}}{\beta^{(t,0)}}$ is γ_β^2, when $\beta^{(t,k)} = \gamma_\beta$, and $\beta^{(t,0)} = \frac{1}{\gamma_\beta}$. Accordingly, the minimum of $\frac{\beta^{(t,k)}}{\beta^{(t,0)}}$ is $\frac{1}{\gamma_\beta^2}$. We can derive the upper and lower bound also for $\|a_m\|_1$ and $\|a_m\|_2$ as follows:

$$\begin{aligned} \frac{1}{\gamma_\beta^2} \leq\ & a_{m,k} \leq \gamma_\beta^2, \\ \frac{k-1}{\gamma_\beta^2} + 1 \leq\ & \|a_m\|_1 \leq (k-1)\gamma_\beta^2 + 1, \\ \frac{k-1}{\gamma_\beta^4} + 1 \leq\ & \|a_m\|_2 \leq (k-1)\gamma_\beta^4 + 1, \\ & \frac{\|a_m\|_2}{\|a_m\|_1} \leq \gamma_\beta^2, \end{aligned} \quad (27)$$

Then, we apply Eq. (26) and (27) to the first item of Eq. (25), and get:

$$O\left(\frac{\alpha^{(t)}}{\sqrt{MkT}}\right) \leq O\left(\frac{\gamma_\alpha}{\sqrt{MkT}}\right), \quad (28)$$

Then, we apply Eq. (26) and (27) to Eq. (25) and redefine A B and C:

$$\begin{aligned} A &= \alpha^t \sum_{m=1}^{M} \frac{\|a_m\|_2^2}{M\|a_m\|_1^2} \\ &\leq \gamma_\alpha \sum_{m=1}^{M} \frac{\|a_m\|_2^2}{M\|a_m\|_1^2} \\ &\leq \gamma_\alpha \sum_{m=1}^{M} \frac{\gamma_\beta^4}{M}, \end{aligned} \tag{29}$$

$$\begin{aligned} B &= \sum_{m=1}^{M} \frac{1}{M}(\|a_m\|_2^2 - a_{m,-1}^2) \\ &< \sum_{m=1}^{M} \frac{1}{M}[[(k-1)\gamma_\beta^4 + 1]^2 - \frac{1}{\gamma_\beta^2}], \end{aligned} \tag{30}$$

$$\begin{aligned} C &= \max_m \{\|a_m\|_1^2 - \|a_m\|_1 a_{m,-1}\} \\ &< [(k-1)\gamma_\beta^2 + 1]^2 - \frac{1}{\gamma_\beta^2}[\frac{k-1}{\gamma_\beta^2} + 1], \end{aligned} \tag{31}$$

Then, we can combine the first and second items of Eq. (24) and get the new bound:

$$\min_{t\in[T]} \mathbb{E}\|\nabla F(W^t)\|^2 \leq O\left(\frac{P}{\sqrt{MkT}} + \frac{Q}{kT}\right), \tag{32}$$

where P is defined by:

$$\begin{aligned} P &= \gamma_\alpha + A\sigma^2 \\ &= (\sum_{m=1}^{M} \frac{\gamma_\beta^4 \sigma^2}{M} + 1)\gamma_\alpha, \end{aligned} \tag{33}$$

and Q is defined by:

$$\begin{aligned} Q &= MB\sigma^2 + MC\rho^2 \\ &= \sum_{m=1}^{M} [[(k-1)\gamma_\beta^4 + 1]^2 - \frac{1}{\gamma_\beta^2}]\sigma^2 + M\rho^2[[(k-1)\gamma_\beta^2 + 1]^2 - \frac{1}{\gamma_\beta^2}[\frac{k-1}{\gamma_\beta^2} + 1]]. \end{aligned} \tag{34}$$

Note that we use the upper bound of A, B, C here. Now we have completed the proof of Theorem 1.

B Supplementary Experimental Results

B.1 Learning Rate Curve of FedHyper

As we analyzed in Section 3, FedHyper adjusts the learning rates in a way that the learning rates increase in the former training stages and decrease in the latter stages. It also aligns with our analysis of the relationship between the convergence and the value of learning rates in Figure 1. To illustrate this point, we visualize the change in global learning rate through 0-100 epochs in Figure 6. The global learning rate of FedHyper-G increases from round 0 to 15, and starts to decrease. The value is greater than 1 in the first 50 epochs and less than 1 in the following 50 epochs. As for FedExp, the global learning rate value fluctuates between 1.0 and 1.5.

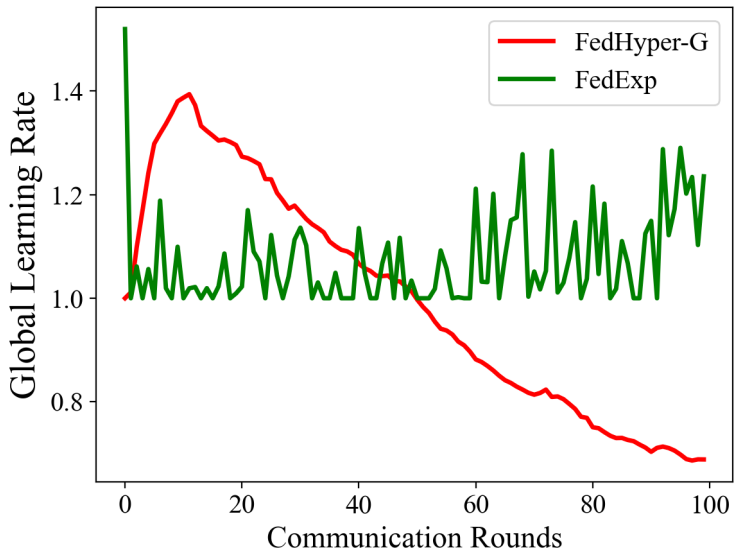

Figure 6: Comparison of global learning rate curve between FEDHYPER and FEDEXP.

B.2 THE TIME OVERHEAD OF FEDHYPER

FEDHYPER involves utilizing the inner product of gradients to modify the learning rate. It may raise concern that FEDHYPER incurs additional computational expenses compared to the more basic FedAvg. Here we claim that hypergradient computing in FEDHYPER only involves a small amount of time overhead, which is trivial compared to the original time consumption of FEDAVG. To prove this, we have included an analysis of the time cost of our three schedulers and baselines. We ran the experiments on time cost of our three schedulers and baselines. The following results in Table 1 are the time cost (in seconds) of different optimization algorithms when training CIFAR10 for 200 communication rounds:

FedHyper-G	FedHyper-CL	FedAvg	FedExp	FedAdam	FedAdagrad
44060	45780	44031	45193	44045	43917

Table 1: Total time consumption in second of FEDHYPER and baselines

Different global schedulers have similar running times, however, the schedulers in FEDHYPER can reach convergence faster than baselines, so we can have less cost than our baselines ultimately. Specifically, FEDHYPER-CL only increases less than 5% computation cost compared with FEDAVG(SGD) but gets convergence up to 3 times faster than SGD. Therefore, FEDHYPER's faster convergence offsets this slight increase in computation time.

B.3 THE IMPACT OF NON-IID LEVEL

The data distribution on clients also affects FL performance. In Table 2, we display the final accuracy of global models with FEDAVG and FEDHYPER in iid and non-iid data, and also different α in non-iid Dirichlet distribution. We can conclude from the table that FEDHYPER contributes more to non-iid settings, especially with relatively small α numbers. The accuracy increase of $\alpha = 0.25$ is 0.80% in FMNIST and 1.55% in CIFAR10, which is the highest among the three α values. This might be attributed to the client-side local scheduler we designed that adopts the global updates to restrict the increasing of local learning rates on some clients that might hinder the global convergence because of the data heterogi.

	FMNIST				CIFAR10			
	IID	0.75	0.5	0.25	IID	0.75	0.5	0.25
FedAvg	98.62%	96.35%	96.36%	97.00%	67.14%	56.71%	57.14%	53.55%
FedHyper	98.92%	97.03%	96.97%	97.80%	67.45%	57.18%	58.42%	55.10%

Table 2: Accuracy on FedHyper in different α values

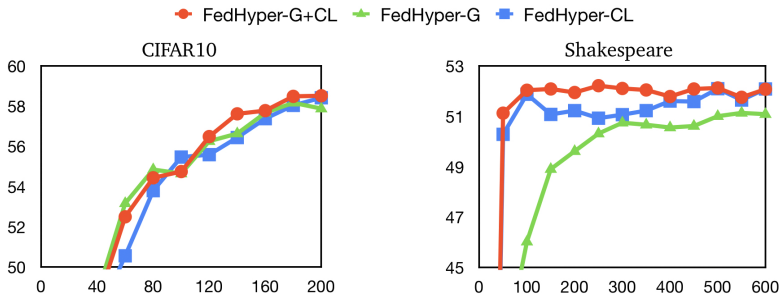

Figure 7: Cooperation of FEDHYPER-G and FEDHYPER-CL.

B.4 COMBINATION OF FEDHYPER-G AND FEDHYPER-CL

We show the results of FEDHYPER-G, FEDHYPER-SL, and FEDHYPER-CL work alone in Figure 3 and show that they can both outperform baselines that optimize the global or local training from the same dimension (e.g. both work on the server). However, FEDHYPER has another advantage over baselines, that is, the ability to adjust both global and local learning rates in one training process. To support this, we run experiments on CIFAR10 and Shakespeare by applying both FEDHYPER-G and FEDHYPER-CL, called FEDHYPER-G+CL. We display the results in Figure 7 and compare it with FEDHYPER-G and FEDHYPER-CL. The results show that FEDHYPER-G+CL is still able to outperform both of the single adjusting algorithms, which indicates that the performance of FEDHYPER algorithms can superpose each other. This makes FEDHYPER more flexible and can be suitable for different user needs. Here we do not show the results of combining FEDHYPER-G and FEDHYPER-SL because they use the same hypergradient to adjust different learning rates. So the superposition effect is not obvious.

B.5 HOW TO USE FEDHYPER IN REAL FL PROJECTS

We have three schedulers in FEDHYPER framework. However, not all of them are needed in real FL projects. We highly recommend FL trainers select suitable algorithms according to their needs and budgets. Here we provide some suggestions on the algorithm selection in specific scenarios.

FedHyper-G only when the trainer has a tight budget of computational resources on clients, e.g., when performing FL on edge devices, mobile terminal devices, or low-memory GPUs.

FedHyper-SL only has a similar scenario with FEDHYPER-G only. One thing difference is that it adds some extra communication costs in sending the local learning rates. Therefore, if the trainer does not have a bottleneck in communication cost, she can choose freely between FEDHYPER-G only and FEDHYPER-G only while considering the features of the specific task (i.e., more sensitive to global or local learning rates).

FedHyper-CL only when the trainer has a tight budget of computational resources on the server but a loose budget on clients, e.g., FL service providers.

FedHyper-G and FedHyper-CL when the trainer has loose budgets of computational resources on both server and clients, e.g. distributed large model training.

Algorithm 1 Workflow of FEDHYPER

Input: Initial Global Model W^0, Number of Communication Rounds T, Number of Selected Clients each Round M, Initial Global Learning Rate α^0, Initial Local Learning Rate each Round $\beta^0 = \beta^1 = \beta^0 = ... = \beta^{T-1}$, Local Epoch number K, Local batches ξ;
Output: Trained Global Model W^T;
for t in $0, 1, ..., T-1$ **do**
  Server send W^t and β^t to all selected clients.
Clients: FedHyper-CL
  Compute global model update $\Delta^{t-1} = W^t - W^{t-1}$
  $\beta_m^{t,0} = \beta^t$
  for k in $0, 1, ..., K-1$ **do**
    Compute $g_m(W^{t,k})$ on $W^{t,k}$ and ξ,
    Update local learning rate: $\beta_m^{t,k} = \beta_m^{t,k-1} + g_m(W^{t,k}) \cdot g_m(W^{t,k-1}) \cdot (1 + \varepsilon \cdot \frac{g_m(W^{t,k}) \cdot \Delta^{t-1}}{|g_m(W^{t,k}) \cdot \Delta^{t-1}|})$
    Clip: $\beta_m^{t,k} = \min\{\max\{\beta_m^{t,k}, \frac{1}{\gamma_\beta}\}, \gamma_\beta\}$
    Local model update: $W_m^{k+1} = W_m^k - \beta_m^{t,k} \cdot g_m(W^{t,k})$
  end for
  Send $\Delta_m^t = W_m^{t,K} - W^t$ to server.
Server: FedHyper-G
  Compute global model update: $\Delta^t = \Sigma P_m \Delta_m^t$
  Update global learning rate: $\alpha^t = \alpha^{t-1} + \Delta^t \cdot \Delta^{t-1}$
  Clip: $\alpha_t = \min\{\max\{\alpha_t, \frac{1}{\gamma_\alpha}\}, \gamma_\alpha\}$
  Update global model: $W^{t+1} = W^t - \alpha^t \cdot \Delta^t$
end for

We do not encourage other combinations because we do not observe an obvious performance improvement on them. We believe that FEDHYPER-G + FEDHYPER-CL can achieve the best performance in our framework if the trainer has a generous budget.

C ALGORITHM OF FEDHYPER

We obtain the full FEDHYPER algorithm and show the cooperation of FEDHYPER-G and FEDHYPER-CL in Algorithm 1. Note that FEDHYPER-SL uses the same hypergradient as FEDHYPER-G so it is not applied in order to simplify the algorithm.