TEMPO: IMPROVING TRAINING PERFORMANCE IN CROSS-SILO FEDERATED LEARNING

Chen Ying[⋆] *Baochun Li*[⋆] *Bo Li*[†]

[⋆]University of Toronto [†]Hong Kong University of Science and Technology

ABSTRACT

Different from its commonly studied scenario to centrally store clients' data in institutions, which implicitly neglects clients' data privacy, we study cross-silo federated learning in a preferable setting to keep private data on clients, and train the global model with a three-layer structure, where the institutions aggregate model updates from their clients for several rounds before sending their aggregated updates to the central server. In this context, we mathematically prove that the number of clients' local training epochs affects the global model performance and thus propose a new approach, *Tempo*, to adaptively tune the epoch number of each client through training. The results of our evaluation conducted under real network environments show that *Tempo* can not only improve training performance in terms of global model accuracy and communication efficiency, but also the elapsed training time.

***Index Terms*—** Federated learning, cross-silo federated learning, edge computing, non-IID data

1. INTRODUCTION

As a particular branch of federated learning (FL), cross-silo FL [1] has been proposed for the scenario where different *institutions* (*e.g.,* financial or medical organizations [2–5]) or geo-distributed datacenters jointly train a global model with their siloed data. FL is used in this context since such institutional data should not be shared directly due to confidential or legal constraints.

In previous work, cross-silo FL is commonly described as a two-layer structure with one central server and several institutions. It implicitly assumes that each institution centrally stores raw data from a massive number of clients within the institution itself. Each client, likely an edge device, generates large volumes of data. During one training iteration, every institution computes an update of the current global model maintained by the central server, and only communicates this update with the central server.

Within the two-layer structure mentioned above, the data privacy of each institution can be well-preserved. However, in financial and medical industries where sensitive user data needs to be kept private, it is not desirable to store clients' raw data centrally within the same institution from a security point of view. In addition, from a performance point of view, sending raw data from a client to its institution may take a longer time than sending model updates. This motivates a three-layer structure, where traditional FL is used *both* within the institution *and* across institution silos.

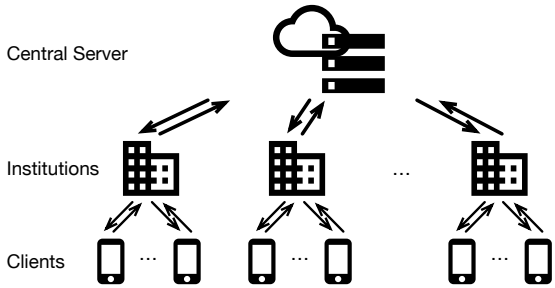

Fig. 1: A three-layer structure of cross-silo FL.

Therefore, we study such a new and more realistic context of cross-silo FL, with a three-layer structure shown in Fig. 1. Clients on the bottom layer conduct local training and send their model updates to their institutions. Institutions on the middle layer aggregate updates from their clients and transfer their aggregated updates to the central server on the top layer. The central server then aggregates updates from institutions to construct an improved global model.

Naturally, cross-silo FL with three layers inherits and magnifies all the inherent challenges in FL: limited and unstable network connections, a massive number of clients, and non-IID (not independent and identically distributed) distributions of clients' local datasets [6]. While the first two problems would result in substantial communication overhead and training time, the last one affects the global model performance, as it has been proved that non-IID data could significantly reduce the accuracy and slow down the convergence of the global model [7, 8].

Although these challenges in FL have been widely studied [9–12], most existing work only considered cross-device FL that involves a central server and numerous mobile devices as clients [8, 13, 14]. Some discussed cross-silo FL, but with institutions centrally storing clients' data [15, 16]. Since three-layer cross-silo FL has a more complex model training scheme, improving training performance under this rarely studied structure would be more challenging.

In this paper, we seek to improve the training performance of cross-silo FL with three layers. Highlights of our **original contributions** are as follows. *Firstly*, to the best of our knowledge, we are the first to study the problem of improving training performance in cross-silo FL with three layers. *Secondly*, our mathematical analysis indicates that epoch numbers of clients' local training largely affect the global model performance and thus propose *Tempo*, a new framework to adaptively tune clients' epoch numbers based on the differences between their institutions' aggregated models and the current global model. *Finally*, we evaluate *Tempo* under real network environments. The results show that *Tempo* can increase the global model accuracy while reducing the number of required training iterations and the elapsed training time.

2. CROSS-SILO FEDERATED LEARNING WITH THREE LAYERS

2.1. System Model

To formulate the model of three-layer cross-silo FL, we suppose that there are N clients and M institutions in total. Each client n, $n \in [N] = \{1, 2, \ldots, N\}$, has a local dataset $\mathcal{D}_n$, which is a set of training samples $\{\mathbf{x}, y\}$ following the distribution p_n. If this client n belongs to the institution m, $m \in [M]$, then $n \in \mathcal{C}^{(m)}$. We define the union of the local datasets of clients of institution m as $\mathcal{D}^{(m)} = \cup_{n \in \mathcal{C}^{(m)}} \mathcal{D}_n$, and the union of all local datasets as $\mathcal{D} = \cup_{m=1}^{M} \mathcal{D}^{(m)}$.

Here we consider a classification problem of S classes. The problem is defined over a compact space $\mathcal{X}$ and a label space $\mathcal{Y} = [S]$. Any data point $\{\mathbf{x}, y\}$ of a client's local dataset distributes over $\mathcal{X} \times \mathcal{Y}$. Typically, $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

Same as the conventional two-layer FL, the goal of the three-layer FL is to find global model weights $\mathbf{w}^{(c)} \in \mathbb{R}^d$ that can characterize the output y with a function $f_s(\mathbf{x}, \mathbf{w}^{(c)})$ from the input $\mathbf{x}$, with f_s being the probability for class s. The training process of the global model is to minimize the global loss $F(\mathbf{w}^{(c)})$ based on local datasets. We define $F(\mathbf{w}^{(c)})$ as the widely used cross-entropy loss, which is

$$F(\mathbf{w}^{(c)}) = \sum_{s=1}^{S} p(y=s) \mathbb{E}_{\mathbf{x}|y=s} \big[\log f_s(\mathbf{x}, \mathbf{w}^{(c)}) \big]. \quad (1)$$

If the central server centrally stores all clients' data, $\mathbf{w}^{(c)}$ can be generated by utilizing centralized stochastic gradient descent (SGD). With η as the learning rate, in its t-th iteration, the centralized SGD updates the $\mathbf{w}^{(c)}(t)$ as

$$\begin{aligned} \mathbf{w}^{(c)}(t) &= \mathbf{w}^{(c)}(t-1) - \eta \nabla_{\mathbf{w}^{(c)}} F(\mathbf{w}^{(c)}(t-1)) \\ &= \mathbf{w}^{(c)}(t-1) - \eta \sum_{s=1}^{S} p(y=s) \cdot \\ &\quad \nabla_{\mathbf{w}^{(c)}} \mathbb{E}_{\mathbf{x}|y=s} \big[\log f_s(\mathbf{x}, \mathbf{w}^{(c)}(t-1)) \big], \end{aligned} \quad (2)$$

where p is the data distribution of $\mathcal{D}$.

However, in FL, only clients have access to their local data and can conduct local SGD to iteratively optimize their local models as eq. (2). To compute the global model weights $\mathbf{w}^{(c)}$ in the three-layer cross-silo FL, we modify the widely used FedAvg algorithm [17] originally proposed for two-layer FL.

Suppose that each client conducts local SGD with a local epoch number of τ_c, a minibatch size of B, and a learning rate of η. That is, a client sends its updated weights to its institution after every τ_c local training epochs. In the t-th global training iteration, each institution sends its aggregated weights to the central server after τ_i aggregations. In the t_i-th round of local aggregation at institution m, its client n updates its local model $\mathbf{w}_n$ at its t_c-th local training epoch as:

$$\begin{aligned} \mathbf{w}_n(t, t_i, t_c) = \mathbf{w}_n(t, t_i, t_c - 1) - \eta \sum_{s=1}^{S} p_n(y=s) \cdot \\ \nabla_{\mathbf{w}_n} \mathbb{E}_{\mathbf{x}|y=s} \big[\log f_s(\mathbf{x}, \mathbf{w}_n(t, t_i, t_c - 1)) \big], \end{aligned} \quad (3)$$

where $t_c \in [\tau_c]$ and $t_i \in [\tau_i]$.

After the τ_c-th local training epoch of client n, it sends $\mathbf{w}_n(t, t_i, \tau_c)$ to its institution m. Institution m then updates its aggregated model $\mathbf{w}^{(m)}(t, t_i, \tau_c)$ by aggregating the updated weights from all of its clients:

$$\mathbf{w}^{(m)}(t, t_i, \tau_c) = \sum_{n \in \mathcal{C}^{(m)}} \frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|} \mathbf{w}_n(t, t_i, \tau_c). \quad (4)$$

Before finishing its τ_i-th round of local aggregation, institution m sets its aggregated model at the beginning of its t_i-th round as $\mathbf{w}^{(m)}(t, t_i - 1, \tau_c)$ and sends it to all of its clients. That is, $\forall t_i \in [\tau_i]$ and $\forall n \in \mathcal{C}^{(m)}$,

$$\mathbf{w}_n(t, t_i, 0) = \mathbf{w}^{(m)}(t, t_i, 0) = \mathbf{w}^{(m)}(t, t_i - 1, \tau_c). \quad (5)$$

After τ_i rounds of local aggregation, institution m sends its aggregated model $\mathbf{w}^{(m)}(t, \tau_i, \tau_c)$ to the central server. Then, the central server updates the global model $\mathbf{w}^{(c)}(t, \tau_i, \tau_c)$ by aggregating those aggregated models of all the institutions:

$$\mathbf{w}^{(c)}(t, \tau_i, \tau_c) = \sum_{m=1}^{M} \frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|} \mathbf{w}^{(m)}(t, \tau_i, \tau_c). \quad (6)$$

If this new global model $\mathbf{w}^{(c)}(t, \tau_i, \tau_c)$ satisfies a predefined condition such as convergence, the training is completed; otherwise the $(t+1)$-th iteration of global training begins with the central server sending its current model to all the institutions. That is, $\forall m \in [M]$, we have

$$\mathbf{w}^{(m)}(t+1, 1, 0) = \mathbf{w}^{(c)}(t, \tau_i, \tau_c). \quad (7)$$

2.2. Mathematical Analysis on Trained Global Model

The ideal case for FL is that the trained global model $\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$ computed by eq. (6) is the same as the model

$\mathbf{w}^{(c)}(t)$ in eq. (2), with $t = T\tau_i\tau_c$. Here we use $\mathbf{w}^*(t)$ to replace $\mathbf{w}^{(c)}(t)$ in eq. (2) for clarification and define $\mathbf{w}^*(T, \tau_i, \tau_c) = \mathbf{w}^*(T\tau_i\tau_c)$ for consistency. Hence, the training performance of FL can be quantified by the weight difference between $\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$ and $\mathbf{w}^*(T, \tau_i, \tau_c)$, which we denote as $\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c) = ||\mathbf{w}^{(c)}(T, \tau_i, \tau_c) - \mathbf{w}^*(T, \tau_i, \tau_c)||$.

With the assumption that $\nabla_{\mathbf{w}}\mathbb{E}_{\mathbf{x}|y=s}[\log f_s(\mathbf{x}, \mathbf{w})]$ is $\lambda_{\mathbf{x}|y=s}$-Lipschitz for each class $s \in [S]$, we extend the analysis of two-layer FL in [7] and derive the following inequality to bound $\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$ in three-layer cross-silo FL:

$$
\begin{aligned}
&\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c) \\
&\overset{1}{\leq} \sum_{m=1}^{M} \frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|} \sum_{n\in\mathcal{C}^{(m)}} \frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|} \Bigg(a_n^{\tau_c}\Delta\mathbf{w}^{(m)}(T, \tau_i - 1, \tau_c) \\
&\quad + \eta \sum_{s=1}^{S} ||p_n(y = s) - p(y = s)|| \\
&\quad \sum_{j=0}^{\tau_c - 1} a_n^j g_{\max}(\mathbf{w}^*(T, \tau_i, \tau_c - 1 - j)) \Bigg) \\
&\overset{2}{\leq} \sum_{m=1}^{M} \frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|} \sum_{n\in\mathcal{C}^{(m)}} \frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|} \Bigg(a_n^{\tau_c\tau_i}\Delta\mathbf{w}^{(c)}(T - 1, \tau_i, \tau_c) \\
&\quad + \eta \sum_{s=1}^{S} ||p_n(y = s) - p(y = s)|| \sum_{k=0}^{\tau_i - 1} \sum_{j=0}^{\tau_c - 1} a_n^{j+k\tau_c} \\
&\quad g_{\max}(\mathbf{w}^*(T, \tau_i - k, \tau_c - 1 - j)) \Bigg),
\end{aligned}
\tag{8}
$$

where $\Delta\mathbf{w}^{(m)}(t, t_i, \tau_c) = ||\mathbf{w}^{(m)}(t, t_i, \tau_c) - \mathbf{w}^*(t, t_i, \tau_c)||$, $a_n = 1 + \eta\sum_{s=1}^{S} p_n(y = s)\lambda_{x|y=s}$, and $g_{\max}(\mathbf{w}) = \max_{s=1}^{S} ||\nabla_{\mathbf{w}}\mathbb{E}_{x|y=s} \log f_s(x, \mathbf{w})||$.

From the second inequality in (8), it should be noted that the weight difference after t iterations of global training comes from two parts. One is the weight difference after $(T - 1)$ iterations, *i.e.,* $\Delta\mathbf{w}^{(c)}(T - 1, \tau_i, \tau_c)$, which is amplified by $a_n^{\tau_c\tau_i}$, exponential with τ_c. The other is the probability distance between the data distribution of client n and the distribution of the whole population $\mathcal{D}$, *i.e.,* $\sum_{s=1}^{S} ||p_n(y = s) - p(y = s)||$, which is amplified by $a_n^{j+k\tau_c}$ also related to τ_c.

Moreover, through global training iterations, the effect of τ_c would accumulate on $\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$. If we could calibrate the weight difference in each iteration by adjusting τ_c of each client, it is promising to improve the training performance. Based on this key idea, we propose *Tempo*.

3. TEMPO: DESIGN AND WORKFLOW

3.1. Design

To minimize $\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$, the second term in the second inequality of (8) implies an intuitive solution: tuning τ_c of client n according to $\sum_{s=1}^{S} ||p_n(y = s) - p(y = s)||$. However, since we do not centrally store clients' data, distribution of the whole population p is unknown. Also, due to privacy concern, p_n, the data distribution of client n should not be revealed.

Therefore, we find another direction based on the first inequality in (8). It shows that if we focus on local aggregations at an institution, then after $(\tau_i - 1)$ rounds of local aggregation, $\Delta\mathbf{w}^{(m)}(T, \tau_i - 1, \tau_c)$, the weight difference between an institution's aggregated model and the optimal global model could affect the bound of $\Delta\mathbf{w}^{(c)}(T, \tau_i, \tau_c)$ and it is amplified by $a_n^{\tau_c}$, which is exponential with τ_c.

Hence, we propose *Tempo*, whose main idea is to let the central server tune $\tau_c^{(m)}(t + 1)$, the local epoch number of clients of institution m in the $(t + 1)$-th global training iteration based on $\omega^{(m)}(t) = ||\mathbf{w}^{(m)}(t, \tau_i, \tau_c) - \mathbf{w}^{(c)}(t, \tau_i, \tau_c)||$. The larger $\omega^{(m)}(t)$ is, the smaller $\tau_c^{(m)}(t)$ should be. As we only use $\mathbf{w}^{(m)}$ and $\mathbf{w}^{(c)}$ that are already accessible to the central server, no extra information would be leaked.

Based on our empirical study, setting $\tau_c^{(m)}(t + 1)$ as the following could lead to high training performance:

$$
\begin{aligned}
\tau_c^{(m)}(t + 1) = \Bigg\lceil \frac{c/2}{\min\limits_{i\in[M]} \ln\omega^{(i)}(t) - \max\limits_{j\in[M]} \ln\omega^{(j)}(t)} \cdot \\
\left(3\ln\omega^{(m)}(t) + \min_{i\in[M]} \ln\omega^{(i)}(t) - 4\max_{j\in[M]} \ln\omega^{(j)}(t)\right) \Bigg\rceil,
\end{aligned}
\tag{9}
$$

where c is the epoch number of all clients in the first global training iteration.

3.2. Workflow

In *Tempo*, all institutions follow the same workflow in parallel during every training iteration of the global model. In the t-th global training iteration, the central server, institutions, and clients conduct the following steps.

Step 1: The central server sends weights of the current global model $\mathbf{w}^{(c)}(t, 1, 0)$ with $\tau_c^{(m)}(t)$ to institution m, $\forall m \in [M]$, and the institution m uses this model as its initial aggregated model $\mathbf{w}^{(m)}(t, 1, 0)$. For the first global training iteration, $\tau_c^{(m)}(1) = c, \forall m \in [M]$.

Step 2: At the beginning of its first aggregation, institution m sends model $\mathbf{w}^{(m)}(t, 1, 0)$ and local epoch number $\tau_c^{(m)}(t)$ to its client n, $\forall n \in \mathcal{C}^{(m)}$. For a later t_i-th round of aggregation, institution m only sends $\mathbf{w}^{(m)}(t, t_i, 0)$.

Step 3: Client n, $\forall n \in \mathcal{C}^{(m)}$, conducts local SGD for $\tau_c^{(m)}(t)$ epochs to compute $\mathbf{w}_n(t, t_i, \tau_c^{(m)}(t))$, which is then sent to institution m. Step 2 and 3 are repeated until institution m finishes τ_i rounds of local aggregation.

Step 4: Institution m uploads its aggregated model weights $\mathbf{w}^{(m)}(t, \tau_i, \tau_c^{(m)}(t))$ to the central server. The central server generates an improved global model $\mathbf{w}^{(c)}(t, \tau_i, \tau_c)$ by aggregating aggregated models from all the institutions.

Step 5: The central server computes $\{\tau_c^{(i)}(t+1)\}_{i=1}^{M}$ according to (9). Repeat Step 1 to 5 until the global model converges or reaches a target accuracy or target iteration number.

4. EVALUATION

Experiments of existing work, by and large, deployed the central server, institutions, and clients on the same machine, where computation would be the bottleneck of FL training. However, in the real world, the bottleneck should be communication, but not computation, due to limited and unstable network connections. To provide realistic evaluations, we separately deploy the central server in the public cloud, so that the bandwidth from institutions to the central server is limited and communication would take most of the training time. We deploy the central server on a production server with DigitalOcean, with a Droplet of 1 GB CPU. Institutions and clients are deployed on machines with 4 GPUs of 16 GB.

We evaluate *Tempo* with training popular machine learning models LeNet-5 and ResNet-18 on three widely used benchmark datasets: MNIST, FashionMNIST, and CIFAR-10. The first two datasets use LeNet-5, while the last one uses ResNet-18. As there is no existing method designed for the same problem as we study, we compare *Tempo* with *FedAdp* [18], one of the state-of-the-art methods of two-layer FL. We modify it to a three-layer version by using its aggregating method for both global aggregations on the central server and local aggregations on institutions, and test with three values of τ_c: 4, 6, 8. For *Tempo*, we set $c = 6$.

There are 5 institutions and 200 clients in total. Each institution communicates with 40 clients and $\tau_i = 4$. Each client has 600 samples as its local dataset and conducts local training with a batch size of 10 and a learning rate of 0.01. we study three different data distributions: (1) IID, where clients get independent and identically distributed data samples; (2) non-IID, where data samples are sampled by the symmetric Dirichlet distribution with the concentration parameter of 1; (3) same as (2) but with the concentration parameter of 0.1.

Fig. 2 illustrates the global model accuracy through global training iterations when we use the FashionMNIST dataset and clients have non-IID local datasets with the concentration parameter of 0.1 as an example of our results. This figure indicates that compared with *FedAdp* using any of the three values of τ_c, *Tempo* can converge to the highest accuracy 85.37%. And if the training goal is to each a target accuracy, for instance, 84%, *Tempo* can complete its training within the smallest number of global training iterations, indicating that it can improve communication efficiency as well.

Table 1 presents the elapsed training time to reach a target accuracy under different settings. For each setting, the training time of *FedAdp* shown in the table is the least training time of using a constant τ_c among 4, 6, 8 to reach the target accuracy. Generally, *Tempo* takes less training time than *FedAdp*. Its superiority is more obvious when local datasets

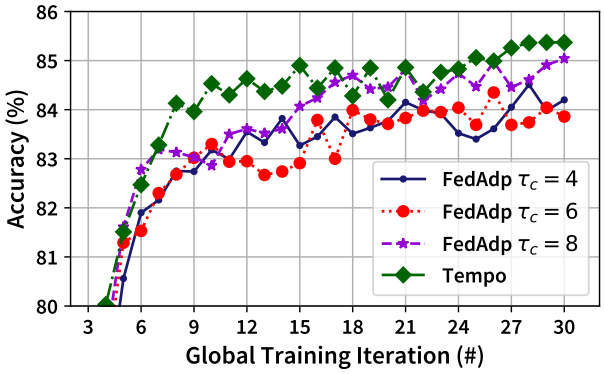

Fig. 2: Global model accuracy of FashionMNIST non-IID local datasets with the concentration parameter of 0.1.

are non-IID with a smaller concentration. This is reasonable since the data distribution of one client will affect its locally trained model and thus affect the aggregated model on this client's intuition. As we use the weight difference between this aggregated model and the global model to tune the epoch number of this client, the weight difference in the next iteration could be indirectly decreased, leading to better training performance.

Table 1: Elapsed training time to reach a target accuracy.

Settings		Training Time (h)	
Dataset	Data Dist., Target Accuracy	*FedAdp*	*Tempo*
MNIST	(1), 98.2%	4.90	4.87
	(2), 96.5%	5.88	5.62
	(3), 92.5%	8.27	7.65
FashionMNIST	(1), 85.9%	7.35	7.34
	(2), 85.5%	8.04	7.93
	(3), 85.0%	10.66	8.82
CIFAR-10	(1), 79.1%	19.13	18.97
	(2), 67.3%	24.45	22.62
	(3), 55.2%	24.76	21.29

5. CONCLUDING REMARKS

In this paper, we focus on improving the training performance of the rarely studied cross-silo FL with three layers. Based on our analysis that the number of local training epochs effectively affects the training performance, we design a new framework, *Tempo*, to adaptively tune the epoch number of each client. The results of our experiments conducted under real network environments indicate that *Tempo* can improve training performance in terms of global model accuracy, communication efficiency, and elapsed training time.

6. REFERENCES

[1] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, et al., "Advances and Open Problems in Federated Learning," *arXiv preprint arXiv:1912.04977*, 2019.

[2] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang, "Federated Tensor Factorization for Computational Phenotyping," in *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 887–895.

[3] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce C. Ho, Li Xiong, and Xiaoqian Jiang, "Privacy-Preserving Tensor Factorization for Collaborative Health Data Analysis," in *Proc. 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019, pp. 1291–1300.

[4] Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, et al., "Deep Learning-Based Classification of Mesothelioma Improves Prediction of Patient Outcome," *Nature Medicine*, vol. 25, no. 10, pp. 1519–1525, October 2019.

[5] Dashan Gao, Ce Ju, Xiguang Wei, Yang Liu, Tianjian Chen, and Qiang Yang, "HHHFL: Hierarchical Heterogeneous Horizontal Federated Learning for Electroencephalography," in *Proc. Int'l Workshop on Federated Machine Learning for User Privacy and Data Confidentiality, in Conjunction with IJCAI 2019 (FL-IJCAI'2019)*, 2019.

[6] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *Proc. Neural Information Processing Systems (NIPS) Workshop on Private Multi-Party Machine Learning (PMPML)*, 2016.

[7] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, "Federated Learning with Non-IID Data," *arXiv preprint arXiv:1806.00582*, 2018.

[8] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, "Robust and Communication-Efficient Federated Learning from Non-iid Data," *IEEE Trans. Neural Networks and Learning Systems*, 2019.

[9] Jakub Konečný, Brendan McMahan, and Daniel Ramage, "Federated Optimization: Distributed Optimization Beyond the Datacenter," in *Proc. 8th NIPS Workshop on Optimization for Machine Learning (OPT)*, 2015.

[10] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni, "Federated Learning with Matched Averaging," in *Proc. 8th International Conference on Learning Representations (ICLR)*, 2020.

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith, "Federated Optimization in Heterogeneous Networks," in *Proc. 3rd Conference on Machine Learning and Systems (MLSys)*, 2020.

[12] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh, "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning," in *Proc. 37th International Conference on Machine Learning (ICML)*, 2020, vol. 119, pp. 5132–5143.

[13] Jun Sun, Tianyi Chen, Georgios B Giannakis, and Zaiyue Yang, "Communication-Efficient Distributed Learning via Lazily Aggregated Quantized Gradients," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[14] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," in *Proc. IEEE INFOCOM*. IEEE, 2020, pp. 1698–1707.

[15] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu, "BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning," in *Proc. USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 493–506.

[16] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang, "Personalized Cross-Silo Federated Learning on Non-IID Data," in *Proc. AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 7865–7873.

[17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, April 2017, vol. 54, pp. 1273–1282.

[18] Hongda Wu and Ping Wang, "Fast-Convergent Federated Learning with Adaptive Weighting," *IEEE Trans. Cognitive Communications and Networking*, 2021.