Automation and Machine Learning (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/autml.2024.050103
ISSN 2516-5003 Vol. 5 Num. 1

Efficient Hierarchical Federated Learning for Unlabeled Edge Devices

Zhipeng Sun[1,2,a,*]

[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China*
[2]*Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, China*
[a]*rodman@mail.ustc.edu.cn*
[*]*Corresponding author*

Keywords: Edge Computing, Hierarchical Federated Learning, Semi-supervised Learning

Abstract: Federated Learning (FL) has emerged as a critical technology to train deep learning models across massive decentralized IoT data on-device. While FL preserves data privacy, it encounters challenges like synchronization latency for model aggregation and single-point failures. In response to these issues, Hierarchical Federated Learning (HFL), which employs edge servers near edge devices to reduce synchronization latency and enhance resilience against single-point failures, has been proposed. However, the assumption of labeled edge devices, i.e, labeled data on edge devices, often proves impractical. Recent researches on semi-supervised FL enable model training for unlabeled edge devices, yet integrating these into HFL presents challenges in balancing model accuracy and training efficiency. This paper introduces FLAGS, a novel semi-supervised HFL system with adaptive global aggregation intervals. Building on the HFL system, FLAGS conducts alternate training between labeled cloud data and unlabeled edge devices. Through an adaptive global aggregation intervals control algorithm, FLAGS navigates the balance between model performance and training efficiency. Evaluation on CIFAR-10 demonstrates FLAGS outperforming baselines within designated time budgets.

1. Introduction

With the proliferation of Internet of Things (IoT) devices generating vast amounts of data, the paradigm of edge computing (EC) [1,2] has emerged as a crucial technology for processing data at its source. EC advances with reduced latency for real-time applications (e.g., autonomous driving, smart city) compared with traditional in-cloud processing. In EC, the advent of Federated Learning (FL) [3] marks a substantial advancement in analyzing and processing distributed data. By collaboratively training deep learning models across multiple clients, FL leverages decentralized data without compromising privacy. The potential of FL has extended across a multitude of applications, including financial services, healthcare informatics, and smart homes [4].

While innovative in facilitating model training at the network edge, FL faces significant challenges, including long synchronization latency during model aggregation and the risk of single-

point failures. To address these limitations, Hierarchical Federated Learning (HFL), which employs edge servers in proximity to edge devices, has been proposed [5]. HFL organizes clients into client sets and model aggregation is carried out in two phases: i) edge aggregation, where locally trained models from clients are uploaded to the edge server to be aggregated into an intermediate model, and ii) global aggregation, where intermediate models are synchronized at the cloud server to produce an updated global model at a certain interval. This dual-phase aggregation offers enhanced training efficiency as well as broadened access to training samples.

HFL assumes labeled data is available on edge devices, which, however, is rarely met in practical scenarios and labeling data can be both time-consuming and costly. Recognizing this challenge, recent research has explored semi-supervised FL (semi-FL) techniques to exploit the abundance of unlabeled data on edge devices. For example, Diao etal. [6] applies Mixup technique to augment the local dataset on clients and pioneer the alternate training phases widely adopted in later literature. Jeong etal. [7] introduce FedMatch, which enforces prediction consistency between edge devices for better performance. Concurrently, Zhao etal. [8] achieve state-of-the-art results by selectively adopting the teacher model (updated as the moving average of the model under training) for pseudo-labeling according to data distribution. Despite the promise shown by these approaches, their integration into HFL systems is challenging, as most of them focus primarily on achieving state-of-the-art model accuracy but overlooks training efficiency—a key objective of HFL.

In this paper, we propose a novel semi-supervised HFL (semi-HFL) system, termed FLAGS, with adaptive global aggregation interval control. FLAGS performs alternate training between labeled data in the cloud and unlabeled data on edge devices, with edge servers facilitating hierarchical model aggregation. The crux of the system design lies in determining global aggregation intervals, i.e., the number of edge aggregations occurring between two global aggregations. On one hand, to provide a solid basis for accurate pseudo labels, smaller global aggregation intervals are preferred to aggregate knowledge from all edge devices and refine the model over in-cloud labeled data for better performance. On the other hand, larger global aggregation intervals amply to the full potential of semi-HFL and contribute to the overall training efficiency. Therefore, our work emphasizes on the critical balance between model performance and training cost, a fundamental concern in practical HFL systems. Evaluation performance demonstrates that FLAGS outperforms existing baselines on the CIFAR-10 dataset by at least 2% within given time budgets.

2. Background

2.1. Federated Learning

We consider an Edge Computing (EC) system composed of three tiers: a central cloud server with robust computing capabilities, M edge servers, and N edge devices or clients. Clients within this system are organized into disjoint sets, with each set associated with an Access Point (AP) that includes an edge server. The training dataset, denoted as $\mathcal{D} = \{x_j, y_j\}_{j=1}^{|\mathcal{D}|}$ comprises $|\mathcal{D}|$ pairs of data samples $\mathbf{x}_j$ and their corresponding labels y_j. In HFL, each client possesses a subset of the data, $\mathcal{D}_i$, and trains a local model $\boldsymbol{w}_i$.

The primary goal of HFL is to obtain an optimal model parameter $\boldsymbol{w}^*$ that minimizes the empirical loss $F(\boldsymbol{w})$. This objective is mathematically formulated as:

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} F_i(\boldsymbol{w}_i) \tag{1}$$

where $F_i(\boldsymbol{w}_i) = \frac{1}{|\mathcal{D}_i|} \sum_{j=1}^{|\mathcal{D}_i|} f(\boldsymbol{x}_j, y_j, \boldsymbol{w}_i)$ is the loss function based on the local dataset on client i.

The HFL process begins with clients conducting initial local training. After a series of local iterations within a global training round, each edge server aggregates the outcomes from its client set to form an intermediate model. The cloud server then periodically collects these intermediate models, amalgamating them into a unified global model, which is subsequently disseminated back down the tiers for further refinement.

2.2. Semi-supervised Hierarchical Federated Learning

Considering the lack of sufficient expert knowledge or labor on clients, it is usually practical that most or even all data on clients are unlabeled while the PS possesses some labeled data annotated by domain experts. Thus, it is appealing to develop Semi-supervised HFL (semi-HFL) building upon the EC system of HFL, where the complete dataset $\mathcal{D}$ consists of labeled dataset $\mathcal{D}_l$ and unlabeled dataset$\mathcal{D}_u$, and $\mathcal{D}_u \triangleq \mathcal{D}_{u,1} \cup \mathcal{D}_{u,2} \cup \cdots \cup \mathcal{D}_{u,N}$ ($\mathcal{D}_{u,i}$ is the dataset of client i). The loss function over a labeled data sample (x, y) and the model parameter $\boldsymbol{w}$ is defined as $\ell_s(x, y, \boldsymbol{w})$. Thus, considering the supervised training on the labeled dataset $\mathcal{D}_l$, the loss function is $\mathbb{E}_{x\in\mathcal{D}_l}\ell_s(x, y, \boldsymbol{w})$.

To leverage the vast amounts of unlabeled data on clients, recent studies have achieved promising results by enforcing model predictions on augmented data that deviate significantly from the data distribution. The objective is to ensure the alignment between these predictions with their corresponding pseudo-labels. In other words, for a given unlabeled data sample x, the model's prediction of its weakly-augmented version is represented as a vector $q = (q_1, \cdots, q_M) \in [0, 1]^M$, where $\sum_{m=1}^{M} q_m = 1$, and M is the number of classes. The pseudo-label for x is then defined as $\hat{q} = \text{argmax}_m q_m$, and is retained only if $\max_m q_m$ falls above the pre-defined confidence threshold τ. Let $\mathcal{H}$ denote the cross-entropy loss function, the unsupervised training loss with consistency regularization can be represented as:

$$\ell_u(x, \boldsymbol{w}) = 1(\max_m (q_m) > \tau)\mathcal{H}(x, \hat{q}, \boldsymbol{w}) \tag{2}$$

Then the total training objective is expressed as:

$$F(\boldsymbol{w}^*) \triangleq \min_{\boldsymbol{w}\in\mathbb{R}^d} [\mathbb{E}_{x\in\mathcal{D}_l}\ell_s(x, y, \boldsymbol{w}) + \mathbb{E}_{x\in\mathcal{D}_u}\ell_u(x, \boldsymbol{w})] \tag{3}$$

3. System Design

3.1. Overview of FLAGS

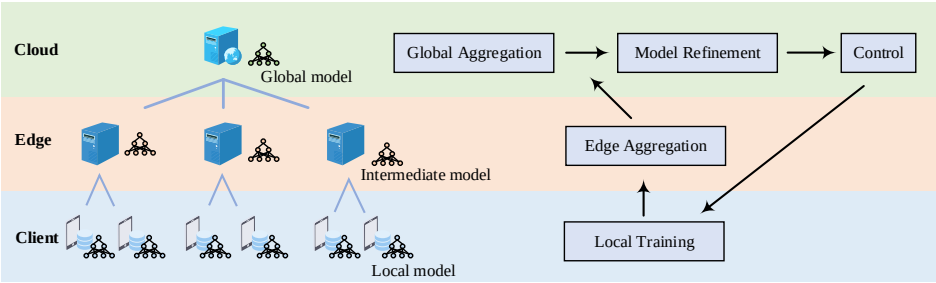

Figure 1: Overview of FLAGS.

The overview of FLAGS's training process is shown in Figure 1. The FLAGS training methodology unfolds with the primary aim of optimizing model accuracy while minimizing training expenses. The process begins with the pretraining of the model on a labeled dataset within the cloud server, setting the stage for subsequent operations. Following this initial step, the pretrained model,

along with the predefined global aggregation interval P, is distributed to edge servers, and the former is further disseminated to a randomly selected subset of clients through the edge server, initiating the model training phase. In this phase, the pretrained model undertakes the task of generating pseudo labels for the unlabeled data present on the client devices. Equipped with these pseudo labels, the clients then engage in training the model for one epoch using both the local data and the generated pseudo labels, applying a consistency loss to enhance the training's effectiveness.

Upon completion of one round, the edge server takes on the role of aggregating the local models furnished by the clients to form an intermediate model. This intermediate model is subsequently redistributed to the clients, marking the beginning of the next round of local training. At the h-th round, the edge server evaluates whether to upload the intermediate model to the cloud for global aggregation, a decision contingent upon the condition P mod h = 0. If the condition is met, the intermediate model is uploaded. Once in the cloud, a global aggregation of the intermediate models received from various edge servers is performed to synthesize a global model. This model then undergoes refinement through training on a limited set of labeled data available in the cloud, further enhancing its accuracy and reliability. This cyclical process of local and global training iterations, punctuated by strategic model aggregations, completes the overall training process of FLAGS.

3.2. System Control

Considering diverse cost preferences for various tasks, such as rapid convergence or minimal communication costs, we define a weighted function encompassing time and communication costs as follows:

$$\Phi_h = \alpha t_h + (1-\alpha) b_h \tag{4}$$

where t_h and b_h donate the time and communication cost in round h respectively and $\alpha \in [0,1]$ adjusts the preference between the two costs. Given the intricate factors influencing model training in HFL (e.g., models, datasets, and the number of clients and edge server), pre-determining the optimal global aggregation interval P is impractical. To navigate this challenge, we introduce a multi-armed bandit (MAB) based algorithm for adaptively determining P. This MAB online learning algorithm iteratively selects actions (or "arms") from a set, collects rewards based on these actions, and refines its action-selection strategy based on the observed rewards across rounds.

In optimizing P, we treat different values of P as distinct actions. The decision-making process entails the MAB agent at the cloud server choosing an action each round, then receiving a reward based on the action's outcome. We denote the accuracy of the models $\boldsymbol{w}^{h+1}$ as u_h at the h-th round, which is calculated through a validation set. Then the accuracy change is denoted as Δu_h. The reward of the decision in round h is defined as follows:

$$r_h = \begin{cases} \frac{\Delta u_h}{\Phi_h}, & \text{if } \Delta u_h \geq 0, \\ \Delta u_h \cdot \Phi_h, & \text{otherwise} \end{cases} \tag{5}$$

The formulation of the reward function is crucial for the efficacy of MAB algorithms [9]. Our reward function is designed with dual objectives: enhancing model performance while minimizing training costs. Firstly, when models achieve identical improvements in accuracy (*i.e.*, $\Delta u_h \geq 0$), those achieved with lower training costs are rewarded more generously. This aligns with our goal of achieving substantial accuracy improvements at minimal costs. Secondly, in instances where certain actions may lead to a decrease in accuracy (*i.e.*, $\Delta u_h < 0$), we still employ $\Delta u_h / \Phi_h$. However, a reduced training cost in such scenarios results in a larger penalty, where the reward is negative. This discrepancy is incongruent with our design principle of fostering efficient training. In such situations, we designate the reward as $\Delta u_h \cdot \Phi_h$.

Typically, the true reward of an action in MAB algorithms is estimated by averaging rewards across rounds. However, this approach does not suit our context for several reasons. Model accuracy improvement rates vary throughout the training process, generally faster in the early stages and slowing over time. Additionally, the optimal action may shift as the training progresses due to improvements in pseudo-label quality and evolving cost preferences, highlighting the non-stationary nature of our MAB problem [10]. To address this, we assign greater weight to more recent rewards and decrease the weight of older rewards with a decay factor $\beta \in (0, 1]$. The estimated reward $\hat{r}_{k,a}$ for action a at the h-th round is then formulated as follows:

$$\hat{r}_{h,a} = \begin{cases} \hat{r}_{h-1,a} + \beta(r_h - \hat{r}_{h-1,a}), & \text{if } a_h = a, \\ \hat{r}_{h-1,a}, & \text{otherwise} \end{cases} \tag{6}$$

where a_h represent the action chosen at the h-th round. Our algorithm strikes a delicate equilibrium between exploration and exploitation to maximize the total rewards received. Concretely, exploration investigates various actions to discover potentially superior options, while exploitation focuses on leveraging the known best option. To optimize for the exploration-exploitation trade-off, we employ the Boltzmann exploration strategy [11], which chooses action $a \in S$ at the h-th round under the probability as:

$$p_{h,a} = \frac{e^{\psi \hat{r}_{h,a}}}{\sum_{a' \in S} e^{\psi \hat{r}_{h,a'}}} \tag{7}$$

4. Performance Evaluation

4.1. Simulation Settings

To evaluate the performance of FLAGS, we conduct simulations over an EC system consists of 10 clients, 5 edge servers, and a cloud server. We simulate real-world network conditions with fluctuating network bandwidths as [12]. The entire simulation utilizes the PyTorch deep learning framework for computational tasks with Python's socket library. We adopt the widely recognized dataset CIFAR-10 [13] consisting of 60,000 32x32 color images, which is split into 50,000 training, 9,000 testing, and 1,000 validation images. In our setting, 4,000 out of 50,000 images served as the labeled data in-cloud, and the rest distributed uniformly over clients. For performance evaluation, we train the AlexNet model [14] for 600 rounds (each contains 2 local epochs), coupled with the Stochastic Gradient Descent (SGD) optimizer for parameter updates.

4.2. Simulation Results

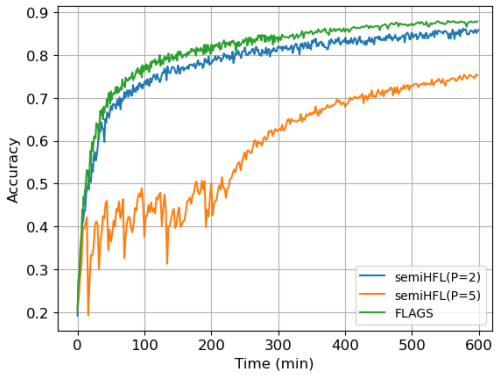

Figure 2: Training Process of FLAGS and baselines on CIFAR-10.

We conduct experiments on CIFAR-10 dataset to evaluate the effectiveness of FLAGS in comparison to baseline models (specifically, semi-HFL with a fixed aggregation interval P). The training progress of these methods is visually depicted in Figure 2. Remarkably, within given time constraints, FLAGS stands out with the fastest convergence rate, surpassing the performance of other models across all three datasets. For instance, under a time budget of 600 minutes, FLAGS achieves an impressive 87.8% accuracy for AlexNet on CIFAR-10. In contrast, semi-HFL with a short aggregation interval, such as P=2, converges quickly but only attains 85.8% accuracy due to substantial time spent on synchronization in the cloud. Conversely, with a longer aggregation interval like P=5, semi-HFL converges poorly, achieving only 75.3% accuracy. These results highlight the superior efficiency of FLAGS in achieving robust convergence and high accuracy.

5. Conclusions

In this paper, we present a novel semi-supervised Hierarchical Federated Learning (semi-HFL) system, termed FLAGS, to explore training models over unlabeled data on edge devices. Based on our analysis of the impact of global aggregation intervals on model performance and training efficiency, we develop a MAB-based algorithm for adaptively determining global aggregation intervals. The experimental results showed that FLAGS outperforms existing baselines by at least 2% on the CIFAR-10 dataset within given time budgets.

References

[1] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing—A key technology towards 5G. ETSI white paper, 11(11), 1-16.

[2] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE internet of things journal, 3(5), 637-646.

[3] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

[4] Aledhari, M., Razzak, R., Parizi, R. M., & Saeed, F. (2020). Federated learning: A survey on enabling technologies, protocols, and applications. IEEE Access, 8, 140699-140725.

[5] Liu, L., Zhang, J., Song, S. H., & Letaief, K. B. (2020, June). Client-edge-cloud hierarchical federated learning. In ICC 2020-2020 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[6] Diao, E., Ding, J., & Tarokh, V. (2022). SemiFL: Semi-supervised federated learning for unlabeled clients with alternate training. Advances in Neural Information Processing Systems, 35, 17871-17884.

[7] Jeong, W., Yoon, J., Yang, E., & Hwang, S. J. (2020, October). Federated Semi-Supervised Learning with Inter-Client Consistency & Disjoint Learning. In International Conference on Learning Representations.

[8] Zhao, J., Ghosh, S., Bharadwaj, A., & Ma, C. Y. (2023). When does the student surpass the teacher? Federated Semi-supervised Learning with Teacher-Student EMA. arXiv preprint arXiv:2301.10114.

[9] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[10] Besbes, O., Gur, Y., & Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. Advances in neural information processing systems, 27.

[11] Cesa-Bianchi, N., Gentile, C., Lugosi, G., & Neu, G. (2017). Boltzmann exploration done right. Advances in neural information processing systems, 30.

[12] Liao, Y., Xu, Y., Xu, H., Yao, Z., Wang, L., & Qiao, C. (2023). Accelerating federated learning with data and model parallelism in edge computing. IEEE/ACM Transactions on Networking.

[13] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Master's thesis, University of Tront.

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.