Adaptive Local Training in Federated Learning

Donald Shenaj
Dept. of Information Engineering
University of Padova
Padova, Italy
dshenaj@gmail.com

Eugene Belilovsky
Concordia University
Mila – Quebec AI Institute
Montreal, Canada
eugene.belilovsky@concordia.ca

Pietro Zanuttigh
Dept. of Information Engineering
University of Padova
Padova, Italy
zanuttigh@dei.unipd.it

***Abstract*—In federated learning multiple clients collaboratively train a global machine learning model by exchanging their locally trained model weights instead of raw data. In the standard setting, every client trains its local model for the same number of epochs. We introduce ALT (Adaptive Local Training), a simple yet effective feedback mechanism that can be introduced on top of any federated learning scheme at the client side to limit unnecessary and degrading computations. ALT dynamically adjusts the number of training epochs for each client based on the similarity between the local representation and the global one, ensuring that well-aligned clients can train longer without experiencing client drift while in case of too large drifts the training is stopped earlier. We evaluated ALT on federated partitions of the CIFAR-10 and Tiny-ImageNet datasets, demonstrating its effectiveness in improving both model convergence speed and accuracy. The code is available at https://github.com/LTTM/ALT.**

***Index Terms*—Federated Learning, Image Classification, Adaptive Training**

I. Introduction

Federated Learning (FL) [1] has emerged as a powerful machine learning paradigm that enables collaborative model training while ensuring data privacy. Unlike traditional centralized approaches, FL allows multiple clients to train a shared model locally, eliminating the need for direct data sharing. This decentralized nature helps to address concerns related to data security, regulatory compliance, and communication costs, making FL particularly well-suited for applications in healthcare, finance, and mobile computing [2].

A key challenge in FL is the heterogeneity of participating clients. Clients may be significantly different in terms of computational power, network bandwidth, and data distribution [3]–[5]. Standard FL algorithms typically train each client for a fixed, pre-defined number of local epochs before aggregating the updates at the server. However, this rigid training schedule does not account for the variability in client resources and data characteristics. In particular, when data distributions across clients are highly heterogeneous, a common scenario in real-world FL applications, training each client for the same fixed number of epochs may lead to severe representation drift in some clients, thereby negatively impacting the global model's convergence and overall performance.

Furthermore, enforcing a uniform number of local training epochs across all clients may lead to inefficient resource utilization. Some clients may be forced to train for longer than necessary, increasing the risk of overfitting on their local data. Others may be unable to complete their assigned training due to resource constraints, leading to incomplete or low-quality model updates. This inefficiency not only affects convergence but also increases energy consumption, which is a critical concern for battery-operated edge devices and contributes to the environmental footprint of large-scale FL deployments.

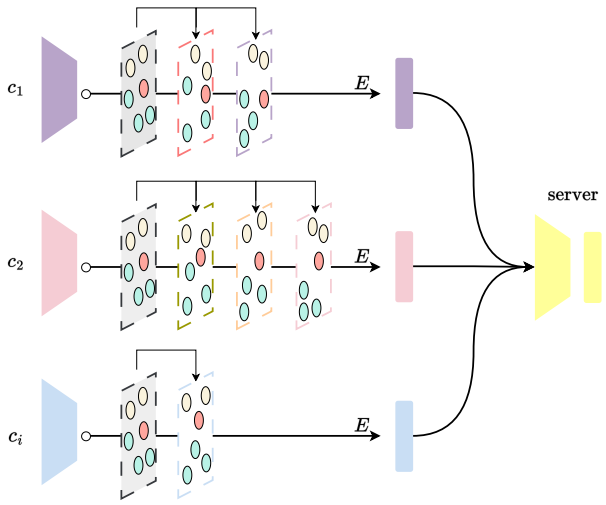


Fig. 1. Overview of the proposed federated learning strategy with dynamically changing local training epochs. At each communication round, clients perform a variable number of local epochs, different between each others, depending on their feature representation. As an example, in the figure the second client performs more local epochs than the other clients.

To address these challenges, we introduce an adaptive training mechanism where each client dynamically determines its number of local training epochs based on a feedback-driven approach (see Figure I). Unlike traditional FL methods that impose a fixed training length, our approach allows the number of local epochs to vary both across clients and across different communication rounds.

Specifically, we propose a client-side control strategy that mitigates representation drift, reduces communication costs, and improves overall FL performance. Our method is based on tracking the evolution of learned representations by measuring the difference in embeddings between the locally trained model and the server-provided global model. By comparing

this difference against a dynamic threshold, each client can autonomously decide when to stop training, ensuring that training is neither excessively prolonged nor prematurely terminated.

Through extensive experimentation, we demonstrate that our proposed adaptive training strategy not only improves model performance but also significantly reduces the total number of local epochs required. This, in turn, leads to lower energy consumption while maintaining or even enhancing the final accuracy of the global model.

The remainder of this paper is organized as follows: Section II reviews related works, while Section III provides a detailed description of the proposed approach. Implementation details are presented in Section IV, followed by experimental results in Section V. Finally, Section VI concludes the paper and outlines future research directions.

II. Related Works

The baseline approach for federated learning is FedAvg [1], where clients independently perform local training on their private data for multiple epochs (differently from local SGD [6]) and then the server aggregates the local models with a simple weighted average. However, it does not allow participating devices to perform variable amounts of local work based on the constraints of their underlying system or on the evolution of the model being learned.

FedProx [7] addresses this issue, assuming that a percentage of the total clients do not complete their local training in a predetermined amount of time. Moreover, this approach introduces a dynamic regularizer in the local objective to limit the impact of local variable updates.

Similarly, SCAFFOLD [3] addresses the client drift by estimating the update directions of both server and clients, to modify client gradients and correct their local updates.

Nonetheless, the interesting results of these methods do not lead to significant improvements with respect to FedAvg when neural networks are deep, as is common for real-world computer vision applications. To this end, MOON [4], proposes a new approach for dealing with non-IID data in presence of deep networks. MOON corrects local training via model-based contrastive learning, enforcing similarity between the current local model and the incoming global one, and dissimilarity between the current local model and the previous local one.

Another dynamic regularization approach is introduced in FedDyn [8] to enable more efficient training in the presence of a large number of devices, unbalanced data, and partial participation. In every federated learning round, the objective of each local device is dynamically updated to ensure that the local objective is asymptotically consistent with stationary points of the global loss.

Dap-FL [9] proposed a a deep deterministic policy gradient assisted adaptive and privacy-preserving FL system, which guarantees that clients with poor resources could participate in FL by adaptively adjusting local training hyper-parameters, and preserves model privacy through a secure aggregation method based on the Paillier cryptosystem. However, the global model prediction accuracy of ResNet-18 on MNIST is not improved with the deployment of the proposed strategy.

Previous approaches disregard internal representations to aggregate model weights. FedMargin [10], instead, computes client deviations based on the margin of class-conditional representations, and uses them to drive federated optimization via an attention mechanism.

Differently from FedProx which accounts for "random" stragglers, in this paper, the internal representation is exploited at the client side to compute the optimal number of training steps, thus limiting client drift. In practice, clients estimate and decide for how long they can train without compromising their representation.

III. Method

A. Federated Learning Setup

Let us assume a set of clients $\mathcal{K}$, where each client $k \in \mathcal{K}$ has access to a local set of samples $(\mathcal{X}_k, \mathcal{Y}_k)$ from a dataset $\mathcal{D}_k$ with $|\mathcal{D}_k| = n_k$. The objective is to collaboratively train a global model $\theta = \{w, v\}$ made of an encoder network w and a decoder v without sharing raw data among clients.

At each communication round $r \in \{1, \dots, R\}$, the server selects a subset of clients $S \subset \mathcal{K}$ to participate in training. Each selected client $s \in S$ initializes its local model θ_s^r with the current global model θ^r and performs local training for E_s^r epochs, where E_s^r is dynamically determined and varies across clients and rounds.

Each client $s \in S$ updates its local model by minimizing the empirical risk over its dataset $\mathcal{D}_s$ using the following loss function:

$$\mathcal{L}_s(\theta) = \frac{1}{|\mathcal{D}_s|} \sum_{(x,y)\in\mathcal{D}_s} \ell(f_\theta(x), y), \tag{1}$$

where $\ell(\cdot,\cdot)$ represents the standard cross-entropy loss for classification tasks, and $f_\theta(x)$ denotes the model's prediction for input x.

During local training, the model parameters are updated using gradient descent on mini-batches $\mathcal{B} \subset \mathcal{D}_s$:

$$\theta_s^r \leftarrow \theta_s^r - \eta \nabla \mathcal{L}_s(\theta_s^r; \mathcal{B}), \tag{2}$$

where η is the learning rate.

After local training, each client sends its updated model θ_s^r to the server. The server then aggregates the received updates using standard federated averaging:

$$\theta^{r+1} = \sum_{s\in S} \frac{n_s}{\sum_{j\in S} n_j} \theta_s^r, \tag{3}$$

where n_s is the number of local samples at client s. This aggregation ensures that updates from larger datasets have a greater influence on the global model.

B. Adaptive Local Training with Embedding Similarity

Traditional FL methods use a fixed number of local epochs E for each client, which can lead to inefficiencies due to heterogeneity in data distributions, computational power, and communication constraints. To address this, we introduce an adaptive mechanism where each client dynamically determines when to stop training based on the similarity between learned representations.

At each training step, we extract feature embeddings from both the local model and the global model. Specifically, let

$$p_s = f(w_s, \mathcal{B}), \quad p_g = f(w_g, \mathcal{B}) \tag{4}$$

denote the feature embeddings produced by the local encoder w_s and the global encoder w_g for the current mini-batch $\mathcal{B}$.

To monitor training progress, we compute the cosine similarity between these embeddings:

$$\cos(p_s, p_g) = \frac{p_s \cdot p_g}{\|p_s\| \|p_g\|}. \tag{5}$$

Local training halts when the similarity condition

$$\cos(p_s, p_g) < T_h(r) \tag{6}$$

is met, where $T_h(r)$ is the value of a dynamically adjusted threshold at round r. This threshold ensures that clients stop training once their local representation becomes too misaligned with respect to the global model received from the server, preventing unnecessary computation and reducing the risk of overfitting. If this condition is never met, the training proceeds until the maximum local epoch limit E is reached. The pseudo-code is reported in Algorithm 1.

C. Benefits of the Adaptive Training Strategy

The proposed adaptive epoch selection method provides several advantages:

- **Efficient Resource Utilization:** Clients train for fewer epochs, reducing energy consumption and extending battery life in mobile devices with limited computational power.
- **Mitigating Representation Drift:** By stopping training dynamically, we reduce the risk of local models diverging too far from the global model, improving overall FL convergence.
- **Faster Convergence:** Adaptive epoch selection prevents overfitting on local data, leading to more stable updates and accelerated global model training.

Note also how the approach introduces only a very limited overhead, i.e., the computation of the embeddings' similarity at the end of each step, thus adding almost no extra complexity to the standard training procedure.

Algorithm 1 Pseudo-code of the proposed ALT Federated Learning algorithm.

Input: Initialize model parameters θ
Initialize maximum rounds R
Initialize threshold parameters a, b

Server executes:
for $r = 1$ **to** R **do**
  $S \leftarrow$ Random subset of clients
  $T_h(r) \leftarrow a + \frac{b*r}{R}$
  for each client $i \in S$ **do**
    $\theta_i \leftarrow$ ClientUpdate$(i, \theta, r, T_h(r))$
  end for
  $\theta \leftarrow \sum_{i \in S} \frac{n_i}{n} \theta_i$
end for

ClientUpdate$(i, \theta, r, T_h(r))$**:**
if $r = 1$:
  $\theta_i \leftarrow \theta$
  $\theta_i := \{w_i, v_i\}$
$\theta_g \leftarrow \theta$
$\theta_g := \{w_g, v_g\}$
stop $\leftarrow False$
for $j = 1, 2, ..., E$ and stop $= False$ **do**
  for each batch $\mathcal{B}$ in $\mathcal{D}_i$ **do**
    $p_i \leftarrow f(w_i, \mathcal{B})$
    $p_g \leftarrow f(w_g, \mathcal{B})$
    if $\cos(p_i, p_g) < T_h(r)$: stop $\leftarrow True$
    $\theta_g \leftarrow \theta_g - \eta \nabla \mathcal{L}(\theta_g; \mathcal{B})$
$\theta_i \leftarrow \theta_g$
return θ_i

D. Thresholding Function Design

A key design choice is how to set the function $T_h(r)$. We experimented different threshold functions $T_h(r)$, including 3 simple approaches:

- Linear Increasing

$$T_h(r) = 0.1 + 0.8 \cdot \frac{r}{R} \tag{7}$$

- Linear Decreasing:

$$T_h(r) = 0.9 - 0.8 \cdot \frac{r}{R} \tag{8}$$

- Fixed (with constant c):

$$T_h(r) = c \tag{9}$$

Among these strategies, the most effective was the linear increasing threshold (7), i.e., $T_h(r) = a + \frac{b \cdot r}{R}$, that linearly increases during the training (we experimentally found that setting $a = 0.1$ and $b = 0.8$ provides the optimal results).

The key insight behind this choice is that the threshold should increase with time, i.e., the criteria should become more and more strict with time, while on the other side the difference w.r.t. the starting model typically increases, up to the point where the condition is satisfied and the training stops.

This formulation enables a "slow start", where early training rounds allow for more flexibility in local updates before gradually enforcing stricter similarity constraints. This progressive tightening helps balance exploration and convergence, leading to improved overall model performance.

Note that, in practice, any thresholding, including a fixed rule (i.e., using Eq. 9), is useful to reduce computation.

IV. Implementation Details

We use PyTorch [11] to implement our approach and the various methods. For CIFAR-10, we use a lightweight CNN as the feature extractor f, consisting of two 5×5 convolutional layers followed by 2×2 max-pooling. The first convolutional layer has 6 channels, while the second has 16. These are followed by two fully connected layers with ReLU activation, containing 120 and 84 units, respectively. For Tiny-ImageNet, we adopt ResNet-50 as the feature extractor, as in [4].

We use the SGD optimizer with a learning rate of 0.01 for all approaches. The weight decay is set to 1×10^{-5}, and the momentum is 0.9. The batch size is fixed at 64. The maximum number of local epochs, E, is set to 10, while the number of communication rounds, R, is 1000 for CIFAR-10 and 150 for Tiny-ImageNet. For MOON, we follow [4], [12] and set the temperature parameter to 0.5 by default.

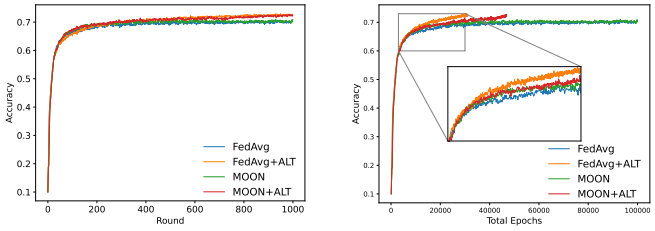
V. Results

We evaluate the performance of our algorithm on the CIFAR-10 [13], and Tiny-ImageNet [14] datasets. CIFAR-10 contains 60,000 32×32 color images belonging to 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. Tiny-ImageNet contains 100,000 color images for training belonging to 200 classes (500 for each class) downsized to 64×64 pixels. Each class has 500 training images, 50 validation images and 50 test images.

As a strong baseline for local training, we consider MOON. We evaluate the performance using $|\mathcal{K}| = 100$ clients, where in each communication round, 10% of the clients are randomly selected to participate in training. The data is partitioned following a Dirichlet distribution [15] with a concentration parameter $\alpha = 100$.

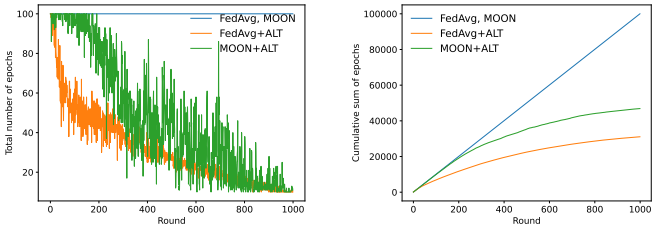
Starting from the CIFAR-10 dataset (Table I and Figure 2), the final accuracy achieved by FedAvg is 70.82%, while by adding our strategy on top of it, it increases to 72.48%.

Similar results can be achieved by applying the proposed approach over the MOON strategy, with an accuracy gain of around 2%. Notice also that FedAvg + ALT can work similarly or better then MOON, while being much more efficient in terms of computational resources.



a) b)

Fig. 2. Accuracy curve on the CIFAR-10 dataset: a) per round; b) per total number of training epochs.



a) b)

Fig. 3. Training length on the CIFAR-10 dataset: a) Total number of epochs performed at each round; b) cumulative sum of the performed epochs during training

However, more than the final accuracy, the key contribution of our approach is allowing to reach a good accuracy with a more limited number of training steps: Figure 3a shows the sum of the local epochs performed by all clients active during each training round, while Figure 3b shows the cumulative sum of training epochs at each round.

While in the standard setting a fixed number of 100 epochs is performed at each round, our approach allows to drastically reduce it, specially when applied on top of FedAvg. Notice how the number of cumulative epochs is reduced to less than a third when applying ALT on top of FedAvg.

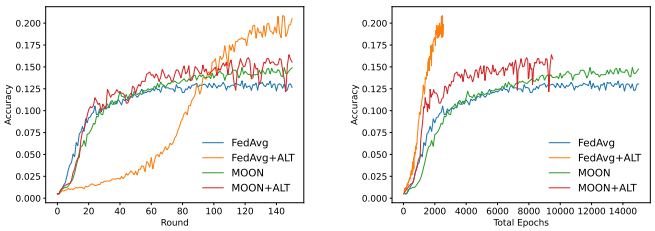
The results on Tiny-ImageNet are shown in Table II while Figure 4 shows the accuracy curves for this dataset. In general, the behavior is similar to the one of CIFAR-10. Our approach, specially when combined with FedAvg allows to reach an higher accuracy, improving the final result from around 12.7% of the standard approach to 20.51% with a gain of almost 8%. When applied on top of MOON the gain is smaller, from 14.89% to 15.56% but still noticeable.

Method	Accuracy	Cumulative Epochs
FedAvg	70.82%	100000
FedAvg+ALT	**72.48%**	**31071**
MOON	70.17%	100000
MOON+ALT	72.14%	46875

TABLE I
Summary of Results on Cifar-10.

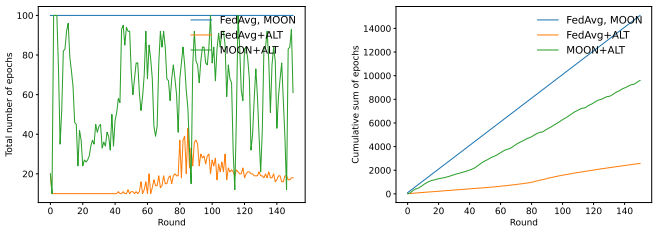
Method	Accuracy	Cumulative Epochs
FedAvg	12.66%	15000
FedAvg+ALT	**20.51%**	**2561**
MOON	14.89%	15000
MOON+ALT	15.56%	9531

TABLE II
Summary of Results on Tiny-ImageNet .



a) b)

Fig. 4. Accuracy curve on Tiny-ImageNet: a) per round; b) per total number of training epochs.



a) b)

Fig. 5. Training length on the Tiny-ImageNet dataset: a) Total number of epochs performed at each round; b) cumulative sum of the performed epochs during training.

Moreover, as for CIFAR-10, the training length is drastically reduced as shown in the plots in Figure 5. In particular the plot in Fig. 5b shows that the cumulative number of epochs is strongly reduced, specially when applying on top of FedAvg. In this case, we achieve a reduction of around 6 times of the number of performed epochs, a reduction bigger than on the other dataset and in general quite impressive. When applying on top of MOON the reduction is smaller but still relevant.

In conclusion, the experiments show that our method allows to reduce substantially computation by reducing the cumulative epochs per round (sum of all clients' epochs), while leading also to improved performance.

VI. Conclusions and Future Works

In this work, we introduced ALT, a novel representation learning feedback mechanism designed to dynamically control the number of local epochs in federated learning. By adaptively adjusting local training duration, ALT effectively reduces both energy consumption and communication costs while maintaining model performance. Our approach is seamlessly integrable into existing FL algorithms without requiring significant modifications.

Experimental results on the CIFAR-10 and Tiny-ImageNet datasets demonstrate that ALT enables more stable training dynamics and accelerates convergence compared to traditional fixed-epoch strategies. The results highlight its potential in enhancing efficiency in real-world FL deployments.

Future research will be devoted to the adaptive setting of other FL parameters and to the evaluation of the proposed approach in combination with other FL strategies.

Acknowledgment

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) Mission 4, Component 2, Investment 1.3, CUP C93C22005250001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, PMLR, 2017.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*, pp. 5132–5143, PMLR, 2020.

[4] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10713–10722, 2021.

[5] D. Shenaj, G. Rizzoli, and P. Zanuttigh, "Federated learning in computer vision," *IEEE Access*, 2023.

[6] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[7] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[8] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *arXiv preprint arXiv:2111.04263*, 2021.

[9] Q. Chen, Z. Wang, J. Chen, H. Yan, and X. Lin, "Dap-fl: Federated learning flourishes by adaptive tuning and secure aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 6, pp. 1923–1941, 2023.

[10] U. Michieli, M. Toldo, and M. Ozay, "Federated learning via attentive margin of semantic feature representations," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1517–1535, 2022.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PmLR, 2020.

[13] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Toronto, ON, Canada, 2009.

[14] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[15] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *European Conference on Computer Vision*, pp. 76–92, Springer, 2020.