# An Element-Wise Weights Aggregation Method for Federated Learning


1st Yi Hu
*Department of Computer Science*
*Swansea University*
Swansea, United Kingdom
845700@swansea.ac.uk

2nd Hanchi Ren*
*Department of Computer Science*
*Swansea University*
Swansea, United Kingdom
hanchi.ren@swansea.ac.uk

3th Chen Hu
*Department of Computer Science*
*Swansea University*
Swansea, United Kingdom
2100552@swansea.ac.uk

4rd Jingjing Deng
*Department of Computer Science*
*Durham University*
Durham, United Kingdom
jingjing.deng@durham.ac.uk

5th Xianghua Xie*
*Department of Computer Science*
*Swansea University*
Swansea, United Kingdom
x.xie@swansea.ac.uk



***Abstract*—Federated learning (FL) is a powerful Machine Learning (ML) paradigm that enables distributed clients to collaboratively learn a shared global model while keeping the data on the original device, thereby preserving privacy. A central challenge in FL is the effective aggregation of local model weights from disparate and potentially unbalanced participating clients. Existing methods often treat each client indiscriminately, applying a single proportion to the entire local model. However, it is empirically advantageous for each weight to be assigned a specific proportion. This paper introduces an innovative Element-Wise Weights Aggregation Method for Federated Learning (EWWA-FL) aimed at optimizing learning performance and accelerating convergence speed. Unlike traditional FL approaches, EWWA-FL aggregates local weights to the global model at the level of individual elements, thereby allowing each participating client to make element-wise contributions to the learning process. By taking into account the unique dataset characteristics of each client, EWWA-FL enhances the robustness of the global model to different datasets while also achieving rapid convergence. The method is flexible enough to employ various weighting strategies. Through comprehensive experiments, we demonstrate the advanced capabilities of EWWA-FL, showing significant improvements in both accuracy and convergence speed across a range of backbones and benchmarks.**



***Index Terms*—Federated Learning, Weights Aggregation, Adaptive Learning**

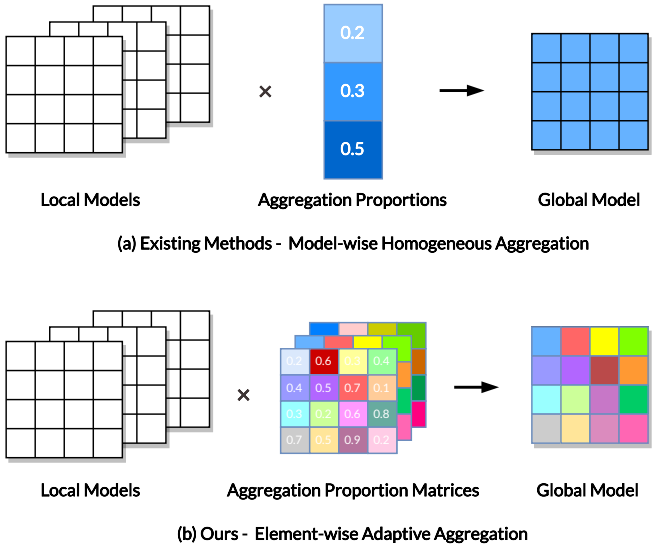




Fig. 1. Illustration for our proposed EWWA-FL

## I. Introduction

As the digital world continues to expand at an unprecedented rate, the world is inundated with a massive amount of data, distributed across various devices, sensors, and platforms. With the growing adoption of Machine Learning (ML) algorithms, the demand for efficient, secure, and decentralized learning processes has become increasingly critical. Federated Learning (FL) [1]–[4] has emerged as a promising solution to address these challenges. It enables the deployment of learning algorithms on decentralized data sources while safeguarding data privacy. FL focuses on training ML models across a multitude of dispersed devices or clients, each holding their own local datasets, eliminating the need for data exchange. This approach effectively addresses privacy and security concerns, as it obviates the need to transfer potentially sensitive data to a centralized location. However, a key challenge in FL lies in the aggregation of model weights. The process of combining model weights from multiple, disparate clients is inherently complex due to the heterogeneous nature of their data distributions [5]–[7]. In an FL network, each client utilizes its local data to train an independent model. Consequently, these local models may capture different data patterns, posing a challenge to the creation of a well-generalized global model. Various strategies, such as Federated Averaging (FedAvg), have been


* The corresponding authors are Hanchi Ren and Xianghua Xie.

developed to mitigate these issues. Nonetheless, devising an efficient and robust aggregation mechanism remains a significant challenge in the field of FL.

There has been a surge in recent research focused on adaptive weight aggregation. Reddi *et al.* [8] proposed a method called FedOpt. They provide a theoretical analysis of the model's convergence on heterogeneous data for non-convex optimization problems, as well as the relationship between dataset heterogeneity and communication efficiency. Three specific optimization methods, termed FedAdam, FedAdagrad, and FedYogi, are employed by the authors. These methods modify the global update rule of FedAvg from one-step SGD to one-step adaptive gradient optimization. Conversely, the work presented in [9] aims to address the challenge of high communication cost in FL. The authors propose a novel communication-efficient adaptive FL method called FedCAMS and also provide a theoretical analysis to guarantee model convergence. They first improve upon FedAdam by incorporating AMSGrad [10] with max stabilization. Both FedOpt and FedCAMS aggregate local updates and obtain an averaged gradient, upon which global model aggregation is conducted. In other words, both methods treat each client equally when generating the global updates. However, the underlying philosophy of adaptive optimization generally favors treating each individual weight independently. While FedOpt and FedCAMS both use adaptive techniques for global model aggregation, their averaging processes do not account for the varied contributions of local models trained on different datasets. This is noteworthy because different datasets result in different levels of convergence [11], [12].

In the work of Federated Boosting (FedBoosting) [11], the authors proposed an adaptive gradient aggregation method based on the boosting algorithm. They discovered that the generalization ability of the global model on Non-Independent and Identically Distributed (non-IID) data is unsatisfactory due to the presence of weight divergence, particularly when employing the FedAvg strategy. Consequently, each client participating in the training receives a unique aggregation percentage. Similarly, Wu *et al.* [12] found that in FL, the path that minimizes the local objective does not necessarily align with the path of global minimization. This implies that each client's contribution to global aggregation will differ. Based on this observation, Wu *et al.* proposed Federated Adaptive Weighting (FedAdp), a method that measures the contributions of participating clients based on the correlation between local and global gradients. All of the above studies have one thing in common: they treat all model parameters equally when aggregating the global model. Specifically, both FedOpt and FedCAMS perform a simple averaging of local model weights prior to subsequent computations. Although FedBoosting and FedAdp assign different proportions to each local model, they still allocate the same proportion to each parameter within these models. This approach may not be the most intuitive or efficient way to handle local models.

Key to the FL process is the merging of model weights from different clients, which is inherently intricate and poses several challenges. The main reason for this complexity is the heterogeneity of the data distribution of the participating entities, cause each client's local dataset has different statistical properties. For example, one client's dataset may contain one or more specific classes, while another client does not. This heterogeneity can result in non-IID data, which poses a significant challenge in aggregating local updates in a way that is representative and conducive to global model performance and generalization. To mitigate the effects of different data distributions and to ensure robust model aggregation, many sophisticated algorithms and techniques are proposed [1], [8], [11], [12]. Building a balanced and harmonious model requires not only rigorous mathematical or algorithmic knowledge, but also a comprehensive understanding of the differences and nuances inherent in the data landscape of different clients. Based on the findings from Wu *et al.* [12], we would like to expand the idea to a more grained level that the elements in each local model have their personalized path to minimize the local objective. It is conceivable that each client should have a different weight, and likewise, each parameter within the local model should also have a unique weight. In the context of convex optimization for learning, the goal is to update the model's weights to achieve convergence. A model comprises various parameters, the values of which fluctuate depending on the feature space of the local dataset. Within the framework of FL, individual local models, trained on distinct datasets, display unique patterns and directions of convergence. As a result, the same parameter across these local models may have vastly different values and may not align closely with each other. Additionally, each parameter may follow a unique trend and orientation toward convergence. As a result, using a uniform proportion to aggregate all parameters into a global model may not be the most suitable approach. Based on this understanding, we introduce Element-Wise Weights Aggregation Method for FL (EWWA-FL), which assigns a different aggregation proportion to each parameter in the local model. Experimental results show that our method outperforms FedAvg, FedCAMS, and FedOpt across various neural networks, benchmark datasets, and experimental settings. The contributions of this work are fourfold:

- We introduce a new perspective on element-wise weight combination for FL. This approach assigns a specific proportion to each parameter in the local model, aiming to improve aggregation. Experimental results confirm the novelty of our proposed EWWA-FL.
- A comprehensive evaluation is conducted. We test the model's generalization ability using various neural networks on different benchmark datasets, employing both Independent and Identically Distributed (IID) and non-IID strategies.
- The adaptive element-wise aggregation paradigm demonstrates faster convergence compared to other recent works.
- We disclose the implementation details of the proposed algorithm to ensure its reproducibility.

The paper is organized as follows: Section II reviews previous studies related to adaptive weight aggregation in FL. Preliminaries on vanilla FL and the Adam optimization algorithm are then discussed. Our proposed approach is elaborated upon in Section III. Section IV provides insights into the experiments, offers in-depth discussions, and suggests potential mitigation methods. Finally, concluding remarks are presented in Section V.

## II. Related Work

A fundamental challenge in FL is the efficient aggregation of model weights from diverse and potentially non-IID data sources to produce a globally consistent model. Adaptive weight aggregation addresses this challenge by assigning different proportions to local model weights based on their quality or relevance, as opposed to treating them equally. This approach recognizes the inherent heterogeneity present in real-world FL environments. It optimizes the performance of the global model by leveraging the more informative weights from local models and potentially mitigates the negative impact of less reliable participants.

In the work [8], the authors provide a comprehensive discussion on adaptive weight aggregation for FL and propose a flexible framework called FedOpt. This framework is capable of incorporating multiple optimization algorithms. The authors specialize FedOpt into FedAdam, FedAdagrad, and FedYogi by employing three example optimization algorithms: Adam [13], Adagrad [14], and YOGI [15]. This approach closely parallels the FedAvg process, diverging only in the final stage of weight aggregation. After obtaining the averaged local gradients, denoted as $\hat{g}$, the first-order momentum matrices $m$ are computed for FedAdam, FedAdagrad, and FedYogi, as detailed in (1). However, the computation of the second-order variance matrices $v$ varies depending on the algorithm. Specifically, FedAdam employs (2), while FedAdagrad and FedYogi utilize (3) and (4), respectively, to derive their second-order matrices.

$$m_r = \beta_1 m_{r-1} + (1-\beta_1)\hat{g}_r \tag{1}$$
$$v_r = \beta_2 v_{r-1} + (1-\beta_2)\hat{g}_r^2 \tag{2}$$
$$v_r = v_{r-1} + \hat{g}_r^2 \tag{3}$$
$$v_r = \beta_2 v_{r-1} + (1-\beta_2)\cdot \hat{g}_r^2 \cdot sign(v_{r-1} - \hat{g}_r^2) \tag{4}$$

where, $r$ is the training round, $\beta_1$ and $\beta_2$ are two momentum parameters, $sign()$ is the symbolic functions. In the end, all those three methods employ Eq.5 for weights aggregation.

$$\omega_r = \omega_{r-1} + \eta_t \frac{m_r}{\sqrt{v_r} + \epsilon} \tag{5}$$

where, $\eta_t$ is the adaptive learning rate, calculated by:

$$\eta_t = \eta_0 \frac{\sqrt{1-\beta_2^r}}{1-\beta_1^r} \tag{6}$$

where $\eta_0$ denotes the initial learning rate, while $\beta_1^r$ and $\beta_2^r$ represent the $r$-th powers of the parameters $\beta_1$ and $\beta_2$, respectively. The authors provide a theoretical analysis to demonstrate the superiority of the proposed FedOpt in comparison to other methods. The primary distinction between FedOpt and our proposed method, EWWA-FL, lies in the location of the optimization algorithm. Specifically, FedOpt employs the optimization algorithm after averaging the local models, whereas EWWA-FL performs the optimization after each local training. As a result, FedOpt treats each client equally and assigns the same aggregation proportion to each local model through averaging. In contrast, our method treats each parameter in every local model differently. Building upon FedOpt, Wang *et al.* [9] introduced FedCAMS with the objective of reducing communication costs. The optimization algorithm in FedCAMS occupies the same position as in FedOpt, thereby ensuring that all local weights are aggregated equally.

Unlike FedOpt and FedCAMS, FedBoosting [11] and FedAdp [12] assign different proportions to each local model to perform adaptive weight aggregation. FedBoosting computes the aggregation proportion based on the results of local training $T_r^i$ and cross-validation $V_r^{i,j}$. The authors first sum all the validation results for a local model across all other local model validation datasets. Then, they calculate the weight of this sum of validation results. Finally, a *Softmax* function is applied to derive the final proportion for each local model. Equations.7, 8, and 9 provide the local weight aggregation proportion $p_r^i$ for the $i$-th local model in training round $r$:

$$p_r^{(i)} = softmax(softmax(T_r^{(i)}) \cdot \sum_{j\neq i}^{N} V_r^{(i,j)}) \tag{7}$$

$$softmax(T_r^{(i)}) = \frac{exp(T_r^{(i)})}{\sum_{j=1}^{N} exp(T_r^{(j)})} \tag{8}$$

$$V_r^{(i,j)} = \begin{pmatrix} V_r^{(1,1)} & V_r^{(1,2)} & \cdots & V_r^{(1,j)} \\ V_r^{(2,1)} & V_r^{(2,2)} & \cdots & V_r^{(2,j)} \\ \vdots & \vdots & \ddots & \vdots \\ V_r^{(i,1)} & V_r^{(i,2)} & \cdots & V_r^{(i,j)} \end{pmatrix} \tag{9}$$

On the other hand, FedAdp focuses on the angle of convergence between the updated local weight and the global weight. In particular, they quantify the contribution of each client in each round of global observations according to the angle $\theta^i$:

$$\theta^{(i)} = arccos(\frac{< G, g^{(i)} >}{||G|| \cdot ||g^{(i)}||}) \tag{10}$$

where $G$ is the global gradient, $< \cdot >$ is the inner product operation and $||\cdot||$ denotes the L2 normalization. To suppress instability caused by instantaneous angular randomness, the angle $\theta_r^i$ is then averaged over previous training rounds $r$:

$$\hat{\theta}_r^{(i)} = \begin{cases} \theta_r^{(i)} & \text{if} \quad r = 1 \\ \frac{r-1}{r}\hat{\theta}_{r-1}^{(i)} + \frac{1}{r}\theta_r^{(i)} & \text{if} \quad r > 1 \end{cases}$$

The authors then designed a non-linear mapping function that quantifies each client's contribution based on angular

information. Inspired by the *Sigmoid* function, they use a variant of *Gompertz* function [16]:

$$\mathcal{F}(\hat{\theta}) = \alpha(1 - \frac{1}{exp(exp(\alpha(1-\hat{\theta})))}) \tag{11}$$

where $\alpha$ is a hyper-parameter. The final proportions for each local model are calculated by giving each client's contribution value into the *Softmax* function.

In comparison to FedOpt and FedCAMS, although FedBoosting and FedAdp provide different aggregation proportions for local clients, their aggregation proportions are still at the model level. In contrast, our proposed EWWA-FL makes progress in this regard by providing a more fine-grained, element-wise aggregation level. This feature enhances the adaptability and convergence of the global model, especially considering that the local models come from different datasets.

## III. Methodology

In this section, we first present a preliminary discussion on FedAvg, as it is the most commonly used method in FL applications. Subsequently, we briefly explain the Adam optimization algorithm, highlighting that Adam provides an adaptive learning rate for each parameter, in contrast to the Stochastic Gradient Descent (SGD) algorithm. Finally, we introduce the EWWA-FL algorithm, which enables element-wise global weight aggregation in FL.

### A. FedAvg

It is the most basic method behind all the recent proposed FL methods. Assuming we have many clients $\mathcal{C}$ and their local datasets $\mathcal{D}$. The task is formulated as $\mathcal{F}$ with weights $\omega$. So the local gradient $g$ is:

$$g_i = \frac{1}{||d_i||}\nabla_\omega \sum_j \mathcal{L}(\mathcal{F}(x^{(j)};\omega), y^{(j)}); \forall i \in ||\mathcal{C}|| \tag{12}$$

Where $||\cdot||$ denotes the L2 normalization of a vector, $x$ and $y$ are the samples and their relevant labels in the $i$-th local dataset $d_i$. The server gathers all the local gradients and conduct the averaging process to generate the next round of global model $\omega_r$. We assume that $||d_i|| = ||d_k||; \forall d_i, d_k \in \mathcal{D}$.

$$\omega_r = \omega_{r-1} - \frac{1}{\mathcal{C}}\sum_{i=1}^{\mathcal{C}} g_i \tag{13}$$

The FedAvg algorithm [17] aims to create a unified model by averaging gradients from various local clients. While this method is effective for centralizing distributed learning, it is not without shortcomings. Specifically, inherent differences in data distributions among clients lead to diverse convergence directions for local model weights. This diversity arises from the incoherent feature spaces of the data, posing challenges for FedAvg. When local datasets differ significantly in their data distributions, this can induce a considerable bias in local model weights. Consequently, the simplistic averaging mechanism employed by FedAvg may not yield optimal results, particularly in the presence of significant data biases or extreme outliers. Recognizing these limitations, we have started exploring more nuanced aggregation strategies, such as weighted averages or other adaptive mechanisms. These approaches gauge the contribution of each local model based on factors like data distribution or its relevance to the overall learning objective [18]–[20]. Employing these refined techniques produces a global model that is both resilient and accurate, skillfully navigating the complexities of differing data distributions and overcoming some limitations inherent to traditional FedAvg methods.

### B. Element-Wise Aggregation for FL

Prior to detail our proposed method, we would like to briefly introduce the Adam algorithm [13] firstly. It is a SGD optimization method based on the momentum idea. Before each iteration, the first-order and second-order moments of the gradient are computed and the sliding average is computed to update the current parameters. This idea combines the ability of the Adagrad [14] algorithm to handle sparse data with the properties of the RMSProp [21] algorithm to deal with non-smooth data. Finally, it achieves very good test performance on both traditional convex optimization problems and deep learning optimization problems. More details of Adam is shown in Algorithm 1.

**Algorithm 1** Adam

**Require:** initial learning rate $\alpha$
**Require:** exponential decay rates $\beta_1, \beta_2 \in [0, 1]$
**Require:** maximum iteration number $I$
initial weights $\omega_0$
initial $1^{st}$-order moment vector $m_0 \leftarrow 0$
initial $2^{nd}$-order moment vector $v_0 \leftarrow 0$
**for** each iteration $i = 1, 2, 3, ..., I$ **do**
  $g_i \leftarrow \nabla_\omega \mathcal{L}(\mathcal{F}(\omega_{i-1}))$ ▶ get gradients at iteration $i$
  $m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1)\cdot g_t$ ▶ update biased $1^{st}$-order moment estimate
  $v_i \leftarrow \beta_2 \cdot m_{i-1} + (1-\beta_2)\cdot (g_t \odot g_t)$ ▶ update biased $2^{nd}$-order moment estimate
  $\hat{m}_i \leftarrow \frac{m_i}{1-\beta_1^i}$ ▶ compute biased-corrected $1^{st}$-order moment estimate
  $\hat{v}_i \leftarrow \frac{v_i}{1-\beta_2^i}$ ▶ compute biased-corrected $2^{nd}$-order moment estimate
  $\omega_i \leftarrow \omega_{i-1} - \frac{\alpha}{\sqrt{\hat{v}_i}+\epsilon}\cdot \hat{m}_i$ ▶ update weights
**end for**
**return** $\omega_I$

All of the above works [8], [9], [11], [12], [17] either average the local models or assign dynamic proportions to the entire local model for global model aggregation. The learning process for deep learning model can be viewed as a convex optimization problem where the weights in the model are trained to reach a minimum point. The basic components of the model are a number of parameters whose values can vary greatly with the different feature spaces of different local datasets. In the FL scenario, local models trained on different local datasets obtain different degrees and directions of convergence. In these local models, the values of the same

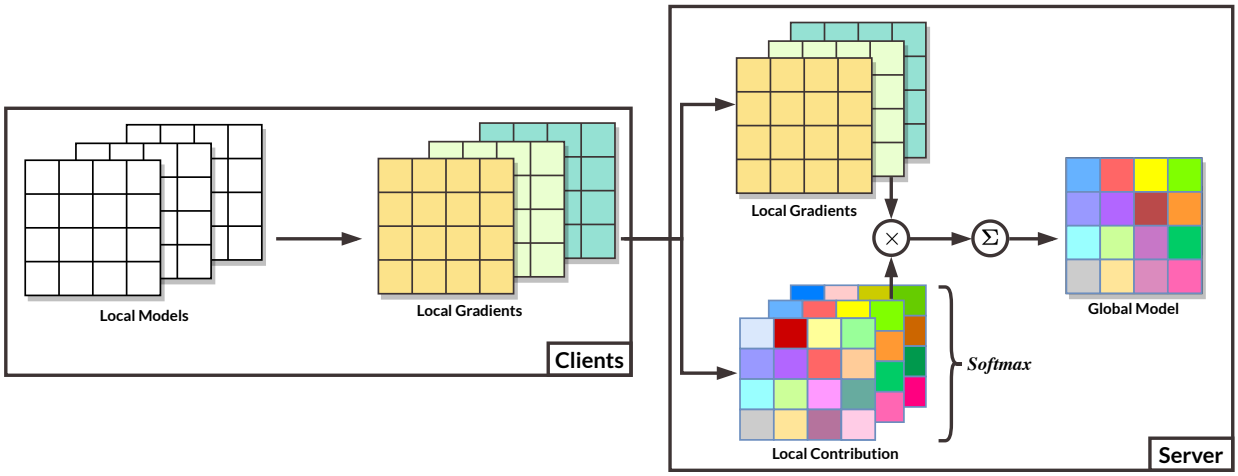



Fig. 2. Diagram of EWWA-FL. The left part is the client that performs the local training. The right side is the server calculating the local contributions and aggregating a new global model based on the local contributions and gradients. All calculations on the server end are done on an element-by-element basis.

parameters may be completely different or even not close to each other, leading to convergence heterogeneity in the local models. Therefore, giving one proportion for all the parameters in the local model is not the best way to aggregate the global model. From another perspective, in contrast to SGD, the adaptive learning methods calculate the learning rate from the point of view of the elements. That is to say, in common model training scenarios, Adam provides an element-wise adaptive learning rate. We would like to follow Adam's idea and introduce element-level adaptive aggregation to FL. As shown in Figure 2, the left part is the client that performs the local training. The right side is the server, which calculates the local contributions and aggregates a new global model based on the local contributions and gradients. All computations on the server end are performed on an element-by-element basis. Specifically, Algorithm 2 is the pseudo-code of our proposed EWWA-FL. In the first round of global training, the global weights $\omega_0$, first-order moment vectors $m_0$ and second-order moment vectors $v_0$ are initialized. Then, the server end assigns weights to each local model for local training. Once the local training is complete, The server collects the gradients of all local models to update the biased first-order moments and second-order moments $m_r$ and $v_r$ and compute the unbiased estimates $\hat{m}_r$ and $\hat{v}_r$. Then obtain the contribution $b_r$ of the local model to the new global model. Finally, the aggregation proportion for each local model is computed using *Softmax*. All the computations related to the first-order moment, second-order moment, local contribution and final aggregation proportion are element-wise, which means each parameter in each local model will receive a specific aggregation proportion in each round, rather than one proportion for all parameters in the local model.

**Algorithm 2** EWWA-FL

**Require:** initial local learning rate $\alpha$
**Require:** exponential decay rates $\beta_1, \beta_2 \in [0, 1]$
**Require:** global training round $R$
initial weights $\omega_0$
initial $1^{st}$-order moment vector $m_0 \leftarrow 0$
initial $2^{nd}$-order moment vector $v_0 \leftarrow 0$
**for** each round $r = 1, 2, ..., R$ **do**
  **for** each client $c \in C$ **do**
    $g_r^{(c)} \leftarrow \nabla_\omega \mathcal{L}(\mathcal{F}(\omega_{r-1}))$
    $m_r^{(c)} \leftarrow \beta_1 \cdot m_{r-1} + (1 - \beta_1) \cdot g_r^{(c)}$
    $v_r^{(c)} \leftarrow \beta_2 \cdot m_{r-1} + (1 - \beta_2) \cdot (g_r^{(c)} \odot g_r^{(c)})$
    $\hat{m}_r^{(c)} \leftarrow \frac{m_r^{(c)}}{1-\beta_1^r}$
    $\hat{m}_r^{(c)} \leftarrow \frac{v_r^{(c)}}{1-\beta_2^r}$
    $b_r^{(c)} \leftarrow \frac{\alpha}{\sqrt{\hat{v}_r^{(c)}}+\epsilon} \cdot \hat{m}_r^{(c)}$
  **end for**
  $p_r^{(c)} \leftarrow \frac{exp(b_r^{(c)})}{\sum_{i=c}^{C} exp(b_r^{(i)})}; \forall c \in C$
  $G_r \leftarrow \sum_{i=c}^{C} p_r^{(i)} \cdot g_r^{(i)}$
**end for**

## IV. Experiments

In this section, we first describe the settings of all experiments. Then, we introduce the backbone neural networks and the datasets for benchmark evaluation. After that, we present multiple sets of experiments to access the performance of our proposed EWWA-FL against other state-of-the-art methods.

### A. Settings, Backbones and Datasets

We utilized the PyTorch framework [22] to implement all neural network models. For anyone interested in replicating our results, the source code is open to the public and can be accessed here[1]. For our proposed EWWA-FL method, the global aggregation learning rate for Adam, Adagrad, and Yogi was set at 1.0 based on the FedOpt reference [8]. We chose the two momentum parameters of 0.9 and 0.999. Local training was conducted using the SGD optimization algorithm,

[1]https://github.com/Rand2AI/EWWA-FL

TABLE I

TOP-1 CLASSIFICATION ACCURACY (%) ACROSS DIFFERENT METHODS, BACKBONE NEURAL NETWORKS AND BENCHMARK DATASETS WITH IID DISTRIBUTION ON LOCAL CLIENTS. "C-10", "C-100" AND "IC-12" STAND FOR CIFAR-10, CIFAR-100 AND ILSVRC2012, RESPECTIVELY. FEDAMS IS DERIVED FROM FEDCAMS AND HAS NO EFFICIENT COMMUNICATION SETTINGS. THE RED ONES ARE THE HIGHEST ACCURACY AND THE BLUE ONES ARE THE NEXT HIGHEST. CROSSED SYMBOLS INDICATE EXPERIMENTAL FAILURE, *i.e.* NO GLOBAL MODEL CONVERGED IN FIVE TRIALS.

| **Model** | | *LeNet* (32*32) | *ResNet-20* (32*32) | | *ResNet-32* (32*32) | | *ResNet-18* (224*224) | | | *ResNet-34* (224*224) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | | MNIST | C-10 | C-100 | C-10 | C-100 | C-10 | C-100 | IC-12 | C-10 | C-100 | IC-12 |
| **FedAvg** | | 98.14 | 91.20 | 58.58 | 91.37 | 61.91 | 89.50 | 68.59 | 65.50 | 89.27 | 68.30 | 67.91 |
| **FedAMS** | | 98.77 | 86.57 | 54.58 | 87.21 | 54.23 | 91.59 | 66.07 | × | 91.91 | 65.41 | × |
| **FedCAMS** | | × | 76.77 | 41.65 | 84.7 | 41.77 | 91.35 | 66.62 | × | 91.44 | 66.41 | × |
| **FedOpt** | **Adam** | × | 73.59 | 22.31 | 74.84 | × | 78.07 | × | × | 81.71 | × | × |
| | **Adagrad** | × | 64.63 | × | 67.36 | × | × | × | × | × | × | × |
| | **YOGI** | × | 65.93 | × | 71.90 | × | 73.88 | × | × | 71.8 | × | × |
| **EWWA-FL** | **Adam** | 98.00 | 89.73 | 64.14 | 90.17 | 65.63 | 90.77 | 70.98 | 65.64 | 91.23 | 70.15 | 68.23 |
| | **Adagrad** | 97.99 | 89.74 | 64.16 | 90.17 | 65.84 | 91.17 | 70.34 | 65.64 | 91.13 | 69.77 | 68.33 |
| | **YOGI** | 98.00 | 89.43 | 64.10 | 90.10 | 65.38 | 90.88 | 70.78 | 65.54 | 91.13 | 70.34 | 68.27 |

TABLE II

PERCENTAGE (%) OF TOP-1 CLASSIFICATION ACCURACY ACROSS METHODS, BACKBONE NEURAL NETWORKS, AND BENCHMARK DATASETS IN NON-IID CONDITIONS ON LOCAL CLIENTS. THE VALUES HIGHLIGHTED IN RED DEMONSTRATE THE HIGHEST PERFORMANCE. THE GLOBAL AGGREGATION OPTIMIZATION ALGORITHM FOR FEDOPT AND EWWA-FL IS ADAM.

| **Model** | *LeNet* (32*32) | *ResNet-20* (32*32) | | *ResNet-32* (32*32) | | *ResNet-18* (224*224) | | | *ResNet-34* (224*224) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | MNIST | C-10 | C-100 | C-10 | C-100 | C-10 | C-100 | IC-12 | C-10 | C-100 | IC-12 |
| **FedAvg** | 96.30 | 72.57 | 55.49 | 75.76 | 57.11 | 82.55 | 66.26 | 50.68 | 82.64 | 65.02 | 46.13 |
| **FedAMS** | 96.78 | 54.78 | 37.33 | 54.46 | 39.71 | 55.71 | 33.71 | 29.26 | 44.13 | 32.12 | 24.09 |
| **FedCAMS** | × | 47.76 | × | 46.47 | 26.87 | 40.03 | 25.57 | × | 41.27 | 24.27 | × |
| **FedOpt** | × | 61.16 | × | 46.75 | × | × | × | × | × | × | × |
| **EWWA-FL** | 96.88 | 76.04 | 56.07 | 78.50 | 57.56 | 83.86 | 66.74 | 51.67 | 84.36 | 65.97 | 52.50 |

accompanied by a consistent learning rate of 0.01 and a momentum of 0.9. The local batch size was set at 64. The ILSVRC2012 dataset was given a training round number of 100, while the other datasets were subjected to 500 rounds. The code we used to reproduce the FedOpt and FedCAMS results was taken from FedOpt's official GitHub repository[2].

The neural networks of choice include *LeNet* [23], *ResNet-18*, *ResNet-20*, *ResNet-32*, and *ResNet-34* [24]. *LeNet* is comprised of three convolutional layers with each succeeded by a *Sigmoid* activation function. The output layer is a fully-connected layer. Its input dimension stands at $32 * 32 * 3$. On the other hand, *ResNet-18* and *ResNet-34* are derivatives from PyTorch's official offerings, having an input dimension of $224*224*3$. As for *ResNet-20* and *ResNet-32*[3], the developers are tailored specifically for the CIFAR-10 and CIFAR-100 datasets [25]. The input size for them is $32 * 32 * 3$.

Specifically, we employed the MNIST dataset [26] for *LeNet*. The CIFAR-10 and CIFAR-100 datasets underwent experimentation using *ResNet-18*, *ResNet-20*, *ResNet-32*, and *ResNet-34*. It is worth noting that for *ResNet-18* and *ResNet-34*, the sample dimensions were upsampling to $224 * 224 * 3$. The ILSVRC2012 dataset [27] was exclusively tested using *ResNet-18* and *ResNet-34*. The datasets were partitioned in a $9 : 1$ ratio for training and testing. Subsequently, the training data is distributed to three local clients, following either an IID or non-IID distribution. The test samples are retained on the server end to assess the performance of the current round of the global model.

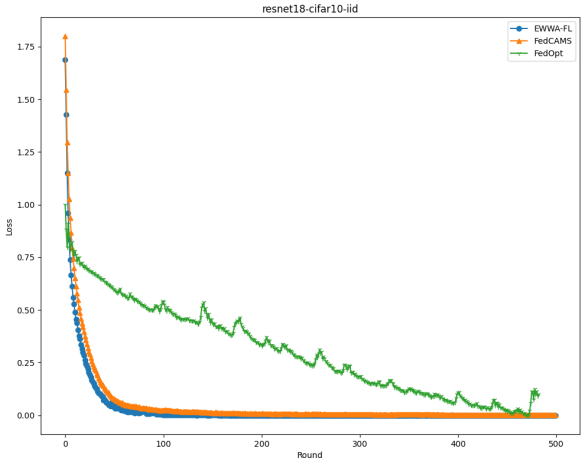
[2]https://github.com/yujiaw98/FedCAMS

[3]https://github.com/akamaster/pytorch_resnet_cifar10

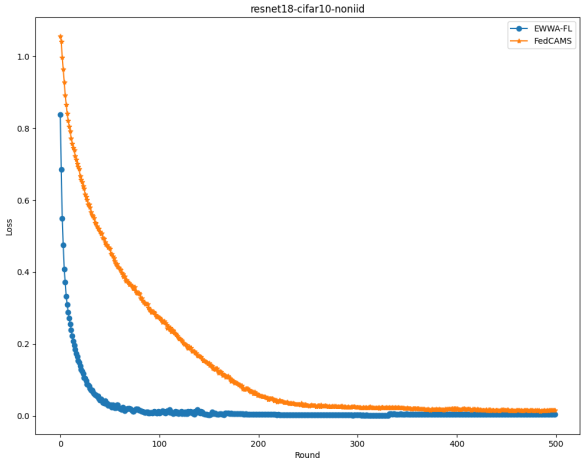
### *B. Accuracy on IID data*

Table I presents a comprehensive comparison of top-1 classification accuracies for various FL methods, employing different backbone neural networks, and tested on multiple benchmark datasets where clients are assumed to have an IID distribution. The table employs color-coding to highlight significant results; values highlighted in red indicate the highest performance for each dataset, while those in blue signify the second highest performance.

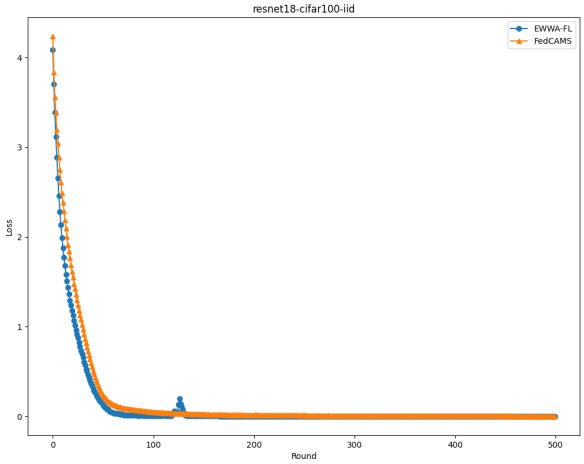
**Performance on Large-Class Datasets:** One of the most noteworthy observations is that our proposed EWWA-FL algorithm exhibits exceptional performance on datasets that have a large number of classes. Specifically, it outshines the competition on the CIFAR-100 and ILSVRC2012 datasets, both of which have a large number of classes, 100 and 1000 respectively. For instance, when employing the *ResNet-20* architecture on the CIFAR-100 dataset, our EWWA-FL model, when optimized using the Adagrad optimizer, achieved an outstanding accuracy of 64.16%. This is considerably better than the next best performing method, FedAvg, which achieved an accuracy of 58.58%. The improvement margin in this case is 9.53%, a significant leap in performance. Similarly, when using *ResNet-32* as the backbone architecture on CIFAR-100, EWWA-FL notched an accuracy of 65.84%,



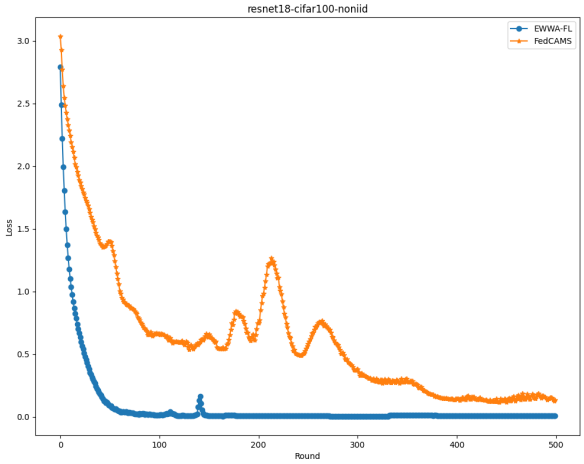

(a) ResNet-18, CIFAR-10, IID



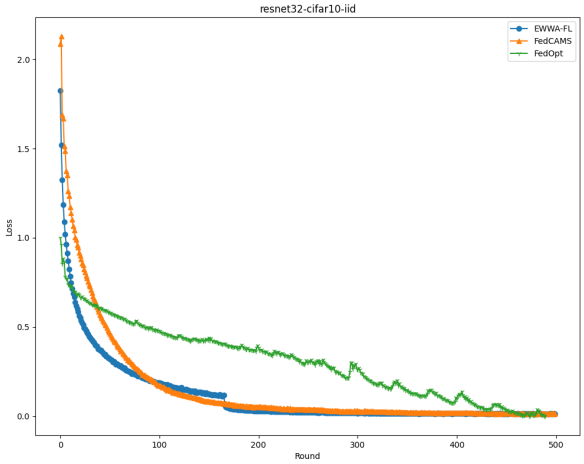

(b) ResNet-18, CIFAR-10, non-IID



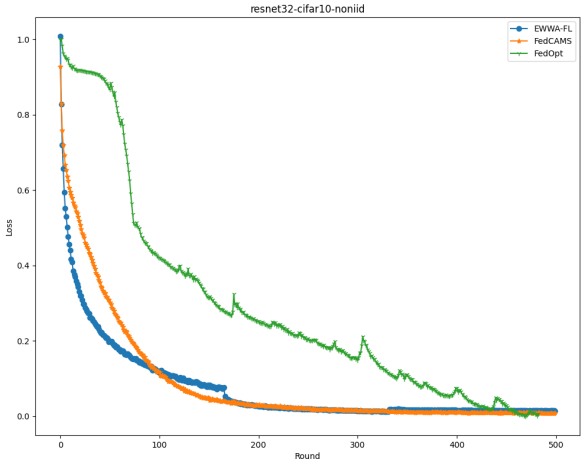

(c) ResNet-18, CIFAR-100, IID




(d) ResNet-18, CIFAR-100, non-IID




(e) ResNet-32, CIFAR-10, IID




(f) ResNet-32, CIFAR-10, non-IID

Fig. 3. Visualization of the average training loss across all local clients is shown. The horizontal axis represents the number of global aggregation rounds, while the vertical axis indicates the average loss for the current round. The blue line represents our proposed EWWA-FL, the orange line is from FedCAMS, and the green line represents FedOpt.

surpassing FedAvg's $61.91\%$ by a margin of $6.35\%$. Though the performance gain is not as high as observed on CIFAR-100, EWWA-FL continues to outperform other methods on ILSVRC2012 dataset as well. For the *ResNet-18* architecture, it reached an accuracy of $65.64\%$, and for *ResNet-34*, it achieved $68.33\%$. Both of these figures are the highest among the tested methods for their respective architectures on this dataset.

**Competitive Results on Smaller-Class Datasets:** Although EWWA-FL does not achieve the highest accuracy on datasets like MNIST and CIFAR-10, it is important to note that the algorithm is highly competitive. For the MNIST dataset, when using the *LeNet* architecture, the highest accuracy was achieved by FedAMS with $98.77\%$. However, EWWA-FL was closely behind with an accuracy of $98.00\%$, making the difference a mere $0.79\%$. On the CIFAR-10 dataset, the performance gaps are also quite narrow. For instance, the differences in accuracy rates when comparing EWWA-FL to the best-performing methods are $1.63\%$, $1.33\%$, $0.46\%$, and $0.75\%$ for architectures *ResNet-20*, *ResNet-32*, *ResNet-18*, and *ResNet-34*, respectively.

In conclusion, our proposed EWWA-FL method exhibits convincing performance, especially in challenging scenarios involving large classes of datasets. Although it is not necessarily the absolute best in all cases, it maintains a competitive edge in various benchmarks.

Compared with FedOpt, that is also capable of employing various global aggregation optimization algorithms, our proposed EWWA-FL significantly outperforms in terms of stability and convergence. In our experiments, the average standard variance for EWWA-FL was remarkably low, at only $0.1094\%$. Additionally, the minimum and maximum variances were confined to a tight range, specifically between $0.0047\%$ and $0.2673\%$, respectively. This suggests that EWWA-FL offers a highly consistent and reliable performance across different scenarios. In contrast, FedOpt showed a much higher variability. The average standard variance for FedOpt was $3.5200\%$, more than thirty times higher than that of EWWA-FL. Furthermore, the minimum and maximum variances for FedOpt were $2.0950\%$ and $4.9550\%$, respectively, indicating a less stable performance. It is worth noting that FedOpt encountered significant issues during our testing phase. Despite conducting at least five separate attempts, none of the FedOpt trials converged as expected. This suggests that FedOpt may have fundamental limitations when it comes to achieving reliable convergence. We trialed code from the official FedCAMS GitHub repository as well as our own deployed code. Similarly, FedAMS and FedCAMS demonstrated a lack of robustness in our experiments. Specifically, FedAMS failed to converge on the ILSVRC2012 dataset, while FedCAMS failed on both the MNIST and ILSVRC2012 datasets. These failures further underscore the superiority of EWWA-FL in achieving stable and consistent results across various benchmark datasets.

### *C. Accuracy on non-IID data*

The circumstance on non-IID data exhibits distinct challenges compared to those on IID data. As the results presented in Table II, our proposed EWWA-FL model consistently outperformed other methods across various benchmarks, including MNIST and CIFAR-10 datasets. For these experiments, We employed Adam as the global optimization algorithm for both FedOpt and EWWA-FL for a fair comparison. Focusing on the MNIST dataset, EWWA-FL achieved an accuracy of $96.88\%$. This figure is marginally but importantly higher by $0.1033\%$ when compared to the $96.78\%$ reported for FedAMS. The small but consistent improvement serves to highlight the efficacy of EWWA-FL in dealing with non-IID data. On the CIFAR-10 dataset, when using *ResNet-20*, EWWA-FL significantly outshone its closest competitor, FedAvg, by achieving an accuracy of $76.04\%$. This was a notable $4.7816\%$ improvement over FedAvg's $72.57\%$. The performance gains extended to more challenging datasets as well. For example, on the ILSVRC2012 dataset, EWWA-FL reached an accuracy of $52.50\%$, which was $13.8088\%$ higher than the $46.13\%$ managed by FedAvg. This suggests that EWWA-FL is not only effective for simpler datasets but also scales well to more complex and larger datasets. In the course of our experimental evaluation, it became evident that certain algorithms like FedAMS, FedCAMS, and FedOpt encountered significant difficulties in reaching convergence. This was consistent with their performance on IID data. Specifically, FedAMS and FedCAMS yielded unsatisfactory results in numerous tests, such as achieving only $37.33\%$ accuracy using *ResNet-20* on the CIFAR-100 dataset and $40.03\%$ accuracy using *ResNet-18* on the CIFAR-10 dataset. Moreover, FedAMS scored as low as $24.09\%$ when tested using *ResNet-34* on the ILSVRC2012 dataset. Similarly, FedOpt struggled in several experiments, underscoring the limitations of current global optimization techniques when dealing with non-IID data distributions. As elaborated in Section III, the model divergence owing to the heterogeneity of local feature spaces is substantial when only a single aggregation proportion is provided for the entire local model. Given that each parameter in the local model can have its own unique direction and level of convergence, the one-size-fits-all approach falls short. In contrast, an element-wise global model aggregation strategy, as implemented in EWWA-FL, offers the adaptability and flexibility needed to facilitate better convergence across a range of neural networks and benchmark datasets.

### *D. Convergence Speed*

Convergence speed is also an important metric for evaluating methods in terms of adaptive aggregation. In Figures 3, we visualize the average training losses across all local clients for both IID and non-IID data. The figures in the left column correspond to experiments conducted on IID data using various backbone neural networks and datasets. Overall, our proposed EWWA-FL method exhibits the fastest convergence when compared with FedCAMS and FedOpt. When using a shallow neural network (*e.g. ResNet-18*), the convergence speeds of

EWWA-FL and FedCAMS are similar. However, the performance gap widens when training on deeper neural network (*e.g. ResNet-32*). On the other hand, in experiments conducted on non-IID data, shown in the right column, the convergence speed of EWWA-FL remains fast. FedCAMS experiences a significant slowdown when using *ResNet-18* on both CIFAR-10 and CIFAR-100 datasets. Nonetheless, FedCAMS achieves good convergence speed when using *ResNet-32* on the CIFAR-10 dataset. We empirically believe that this is because deeper neural networks contain more weights, allowing for better fitting on small-scale datasets. Lastly, FedOpt performs the worst among all the methods in our experiments. The loss values produced by FedOpt are so large that they are difficult to visualize; therefore, we have normalized them to fall within a range of 0 to 1. The green lines indicate that FedOpt struggles to converge, even after 500 rounds of global aggregation. In some experiments, as mentioned in Section IV-B, FedOpt failed to converge. So there is no green lines in Figures 3 (b), (c) and (d).

## V. Conclusion

In this paper, we propose an adaptive element-wise global weights aggregation method for FL, specifically EWWA-FL. This method demonstrates better and faster convergence compared to other recent works. Comprehensive experiments are conducted using various neural networks and datasets to showcase the superiority of our approach. We also provide a brief theoretical analysis based on the Adam optimization algorithm. In future work, we plan to focus on the theoretical proof to further validate our method from a mathematical perspective. The implementation of our method is publicly available to ensure its reproducibility.

## References


[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv*, 2016.

[2] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv*, 2016.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv*, 2016.

[4] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[5] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[6] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "Over-the-air federated learning from heterogeneous data," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3796–3811, 2021.

[7] 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.

[8] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *International Conference on Learning Representations*, 2020.

[9] Y. Wang, L. Lin, and J. Chen, "Communication-efficient adaptive federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 802–22 838.

[10] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv*, 2019.

[11] H. Ren, J. Deng, X. Xie, X. Ma, and Y. Wang, "Fedboosting: Federated learning with gradient protected boosting for text recognition," *arXiv*, pp. arXiv–2007, 2020.

[12] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.

[14] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[15] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[16] M. N. Gibbs and D. J. MacKay, "Variational gaussian process classifiers," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1458–1464, 2000.

[17] 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*. PMLR, 2017, pp. 1273–1282.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv*, 2018.

[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv*, 2019.

[20] C. Xu, Z. Hong, M. Huang, and T. Jiang, "Acceleration of federated learning with alleviated forgetting in local training," *arXiv*, 2022.

[21] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012.

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

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[26] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.