Dynamic Hyper-Parameter Adjustment in Fedprox for Performance Optimization in Federated Learning

Yunfan Zhang *

School of Data Science, The Chinese University of Hong Kong, Shenzhen, China

* Corresponding Author Email: 122090781@link.cuhk.edu.cn

Abstract. This paper investigates the application of dynamic hyper-parameter adjustment strategies to improve the performance of the FedProx algorithm in federated learning (FL). FedProx addresses the challenges posed by data heterogeneity across clients, a common issue in FL that hinders convergence. While FedProx improves upon the basic FedAvg algorithm by introducing a proximal term to stabilize model updates, the hyper-parameter μ that controls the impact of the proximal term is sensitive and needs careful tuning. An inappropriate μ can degrade performance, making dynamic adjustment crucial. The study introduces two dynamic μ adjustment strategies. The first strategy, called ExponentialDecay, progressively reduces μ exponentially by multiplying it with a constant factor. The second strategy, AdaptiveLoss, adjusts μ based on changes in the loss value, decreasing it when the loss decreases and increasing it when the loss increases, offering a more responsive approach to the model's performance. The effectiveness of both strategies is evaluated on the EMNIST dataset, which simulates real-world non-IID data distributions. Results show that both dynamic strategies outperform the static μ setting in terms of model accuracy, loss reduction, and training speed, particularly in scenarios with highly heterogeneous data. Specifically, the AdaptiveLoss strategy achieves the best performance when the data is extremely non-IID, reducing loss by 10.2% and improving accuracy by 2.5% compared to the static μ approach. These dynamic strategies not only enhance FedProx's stability but also accelerate the convergence of the global model.

Keywords: Camera-Ready Paper, Trans Tech Publications, A4 Format.

1. Introduction

With the rapid advancement of smart electronic devices, users' mobile phones and computers have significantly improved in both storage capacity and computational efficiency, enabling them to store more private data and perform higher-complexity tasks directly on their devices. Additionally, the advent of 5G networks has accelerated communication speeds, enhancing the potential for utilizing client data and making it more valuable. In recent years, deep learning-based large models have emerged, showing impressive performance in areas such as image and language processing. However, one of the biggest challenges in training these large models is the need for massive amounts of data. Traditional distributed machine learning approaches address this by collecting raw data from a large number of users' devices to cloud data centers, where model training is performed. While this approach allows models to access large, comprehensive datasets, it introduces significant challenges related to data privacy and communication overhead. Notably, there is no absolute guarantee that users' private data will remain protected during data transfer and storage, leaving it vulnerable to theft and data breaches. Moreover, transmitting large amounts of data over the network consumes substantial bandwidth and requires considerable time, increasing computational costs.

To mitigate these issues, Federated Learning (FL) was introduced. FL is a partially distributed machine learning algorithm that trains models locally on users' devices, only sharing model parameters between the clients and the server. Specifically, FedAvg is a widely-used FL algorithm that treats each user's device as a client and maintains a server to store the global model parameters [1]. In each iteration, the server selects a fraction of clients, sends the global model parameters to them, and each selected client updates its local model using stochastic gradient descent (SGD) based on its local data for a set number of epochs. After local training, the server receives model updates from the clients and aggregates them to update the global model. FedAvg ensures data privacy by

training locally and reduces communication costs by only passing model parameters and performing several local updates. However, this approach faces challenges, particularly with data heterogeneity among clients. Clients' data may vary significantly due to differing usage habits and preferences, causing the training process to diverge from the optimal global solution. This issue has been observed in FedAvg [2].

To address the performance degradation caused by data heterogeneity, the FedProx algorithm was proposed, which modifies FedAvg and shows better performance on highly non-identical datasets [3]. Specifically, FedProx introduces a proximal term in the objective function, which makes client model parameters stay closer to the global model and mitigates the client drift effect caused by local parameter updates. This modification helps ensure convergence and stabilizes the training process. However, FedProx also comes with some disadvantages. The average training speed of FedProx is slower than that of FedAvg due to the additional proximal term. Furthermore, FedProx is highly sensitive to the hyperparameter of the proximal term, denoted as μ. An improper value of μ can significantly reduce performance, making its adjustment a critical task.

This paper investigates methods to improve FedProx by applying dynamic hyperparameter adjustment strategies to the proximal term. The study explores the impact of two dynamic μ adjustment strategies. The first strategy, termed ExponentialDecay, gradually decreases μ exponentially by multiplying it with a constant factor. The second strategy, AdaptiveLoss, adjusts μ based on changes in the loss value—decreasing it when the loss decreases below a threshold and increasing it when the loss increases. Through experiments on the EMNIST dataset, the effectiveness of these dynamic strategies is evaluated, with results showing that they outperform static μ settings in terms of accuracy, loss reduction, and training speed [4].

The remainder of this paper is structured as follows: Section 2 discusses the challenges in federated learning and relevant works. Section 3 presents the methodology and proposed dynamic μ adjustment strategies for FedProx. Section 4 details the experiments and results. Finally, Section 5 concludes the paper.

2. Background and Related Work

Generally, a conventional FL framework aims to utilize the data stored in a huge number of devices to train a Machine Learning model in the global sever [4]. In FL, the Independent Identically Distributed (IID) degree of the data distribution is one of the most important problem environments setting that has a great impact on the performance of a FL algorithm.

Under the situation of statistical homogeneity, which means the data are IID on the client devices, FedAvg shows excellent performance by applying the SGD algorithm in optimizing the model [1]. Unfortunately, the problem in reality is usually heterogenous in many aspects. "In practical large-scale situations, there may be considerable differences between data distributions, model structures, communication networks, and system edge devices, which make it challenging to realize federated collaboration" [5]. FedAvg cannot adapt well to the environment that the data are extremely non-IID among the devices for the following reasons. Firstly, the global model, which is the average of the client models, cannot converge to the optimal solution theoretically when statistical heterogeneity exists. Secondly, large number of local updates on the client models causes the client drift problem. In client drift problem, a local client model often overfit to its training data distribution, making the training process unstable, decreasing the training speed as well as the global model's performance. Thirdly, in each training round of global model, it only randomly selects a small proportion of clients to average, which cannot guarantee the gradient direction to the optimal global solution.

To solve the problem brought by statistical heterogeneity, proposed FedProx and presents its well performance and robustness on several extremely heterogeneous datasets [2]. Based on FedAvg, FedProx add a proximal term to the objective function, which would force the parameter update of the client model approach to the current global model. The proximal term could limit client drift degree and also make the parameter changing of the global model more stable. But there exist some

issues related to the proximal term, in which the obvious one is the hardness of tuning the hyper-parameter μ. The hyper-parameter μ determines the impact of the proximal term in FedProx and a large μ will excessively compel the client model close to the global model and weaken the feature learning process of the client model, leading the decrease of global model performance. In another situation, a small μ will have no effectiveness to limit the overfitting of client model and the FedProx will finally be equivalent to FedAvg when μ decreases to zero. Therefore, a proper μ is crucial for FedProx to have an ideal performance. But still, the presence of proximal term can slow down the training speed of the global model and a proper μ needs an extra number of experiments to find, doubling the overhead of computational resources [2].

To address μ value selecting problem, proposed a research interest, which is dynamically adjusting μ during the training period, trying to find a balance between the training speed and model performance to get the most proper μ in some different stages [2]. Notably, also conducted the experiment of applying a dynamic μ adjustment strategy in FedProx, which showed effectiveness to some degree in accelerating loss decrease and final performance [2]. In this strategy, they increase μ by a constant value when the loss of global model increases and decrease μ by a constant value when the loss of global model decreases continuously for more than a certain rounds.

Except, there also exists other works research the dynamic μ adjustment in different application areas [2]. Sahoo et.al proposed AdaFedProx, improving FedProx framework and applying it to medical image classification to address the privacy problem in this area [6]. Notably, AdaFedProx adopt a reinforcement learning based model - Deep Q-Network (DQN) to dynamically determine the hyper-parameter μ of proximal term for every client. In the end of each round, the testing result, data distributed form, and running efficiency of clients will be regard as the state of current round and input into DQN to compute the value of μ for the next round. AdaFedProx puts several critical information of the model and data into consideration and exhibits better performace than FedAvg, FedProx [1-2]. However, the introducing of DQN determining μ mechanism increase the amount of data size of communication, increase the cost of data transmission. Furthermore, the extra reinforcement learning model DQN expands the computation overhead as well as raise the difficulty of training and generalization, making it hard to prove its effectiveness in different tasks other than medical image area. To address security issues of the Internet of Things (IoT), combined FedProx with Knowledge Distillation (KD), proposing FedKD-Prox to detect IoT invasion [7]. KD enable a simple model to be trained through learning the knowledge distilled from a complex model. As in FedKD-Prox, the introducing of KD also reduces the communication burden and thus, the proximal term in the objective function of FedProx is replaced by the l2-norm of the knowledge value difference between global model and client model. In addition, FedKD-Prox also applies a kind of strategy to dynamically adjust μ based on the knowledge of global and client model. In the experiments, FedKD-Prox was compared with FedAvg and FedProx, showing over all superior performance. Similar to AdaFedProx, the generalization of FedKD-Prox has not been proved in other fields and it is difficult to design proper KD models for all different tasks.

While more other work as and adopted dynamic adjustment idea in FL, they did not research the dynamic μ in FedProx [8-9]. Acar et al proposed FedDyn that introduced a dynamic regularization approach to FL [8]. Haque et al study a method to adaptively tune the hyper-parameter of training epoch number, learning rate, and batch size, tackling the issues bring by device heterogeneity and boosting the effectiveness of model [9].

Inspired by the future work interests of Li et al that mentioned the prospect of studying adaptive μ in FedProx, this paper proposes two strategies of dynamically adjusting μ during the training of FedProx, researching the influence these two strategies could have on the loss decrease and final performance [2]. Specifically, the first strategy keeps reducing μ exponentially and second strategy automatically tunes μ according to the changes of loss value. The experiment was doing on the EMINST dataset, presenting the superiority of dynamic μ adjustment in FedProx under serval circumstances [10].

3. Methodology

This section will first review FedAvg and FedProx frameworks. After that, the two proposed dynamic μ adjustment strategies in FedProx will be explained.

3.1. FedAvg and FedProx frameworks

The objective function in FL can be divided into two parts: global sever and local client. In FedAvg, the objective function of local client is:

$$F_i(\boldsymbol{x}) = \frac{1}{n_i}\sum_{j=1}^{n_i} f(\boldsymbol{x}; \xi_j) \tag{1}$$

Where n_i is the number of data sample ξ_j of client i. Every client model is trained on its local assigned dataset and fit its distribution. The objective function of global sever is:

$$F(\boldsymbol{x}) = \sum_{i=1}^{K}\frac{n_i}{n}F_i(\boldsymbol{x}) = \sum_{i=1}^{K} p_i F_i(\boldsymbol{x}) \tag{2}$$

Where n is the total number of clients and $p_i = \frac{n_i}{n}$ denotes the weight of the client objective function. The global model averages the parameters of client models according to their data size proportion.

While the objective function of global sever for FedProx is the same as FedAvg, the objective function of local client in FedProx is:

$$F_i^{\text{prox}}\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{n_i}\sum_{j=1}^{n_i} f\left(\boldsymbol{x}^{(i)}; \xi_j\right) + \mu|\boldsymbol{x} - \boldsymbol{x}^{(i)}|^2 \tag{3}$$

Where x is the global model parameter, $\boldsymbol{x}^{(i)}$ is the client model parameter, and μ is the hyper-parameter of the proximal term. Correspondingly, the parameter in the local client model will make updates by following:

$$\boldsymbol{x}_{t,j+1}^{(i)} = \boldsymbol{x}_{t,j}^{(i)} - \eta g\left(\boldsymbol{x}_{t,j}^{(i)}, \xi_j\right) - \mu\left(\boldsymbol{x}_t - \boldsymbol{x}_{t,j}^{(i)}\right) \tag{4}$$

Where η is the learning rate and g is the gradient of original objective function for local client in FedAvg. During the training of client models in FedProx, the changes of their parameter will be constrained by the proximal term such that the client model will not be too overfitted to their own dataset.

3.2. Proposed framework: two dynamic μ adjustment strategies in FedProx

FedProx effectively deal with the problems caused by statistical heterogeneity, where the dataset is extremely non-IID on client devices. But the hyper-parameter μ is static during the full training period, which is not suitable for the ever-changing federated model. Specifically, when the gradient of client model leads the global model to an incorrect direction, the hyper-parameter μ should be large to limit this process. In contrast, when the gradient of client model leads the global model to its optimal solution, the constraint of proximal term should be tiny to speed up the training, requiring a relative smaller hyper-parameter μ.

Following the above ideas, two dynamic μ adjustment strategies are designed. The first strategy uses ExponentialDecay tuner to keep decreasing μ exponentially by multiplying a constant factor c. The second strategy uses AdaptiveLoss tuner to decrease hyper-parameter μ exponentially when the loss value decreases lower than a threshold, and increase hyper-parameter μ exponentially when the loss value increases. The propose of designing ExponentialDecay tuner is to limit divergent phenomenon of client model in the early stage and accelerate the training in the late stage [11]. While, the propose of designing AdaptiveLoss tuner is enabling the model to find proper μ values for

different training rounds in a heuristic way. The Algorithm 1 and 2 show the μ adjustment processes of two tuners. In Algorithm 1, μ_{init} is the initial μ value at beginning of train, r is the number of train round, c is a constant between 0 and 1, and μ_{ad} is the adjusted μ value for round r. In Algorithm 2, μ_{prev} is the μ value in the last round, L_{prev} and L_{curr} are the loss values in the last and current round, threshold is a constant value to judge whether μ should be decreased, c_{in} and c_{de} are constant values used to increase and decrease μ, μ_{max} and μ_{min} are constant values used to limit the range of μ, and μ_{ad} is the adjusted μ value.

Algorithm 1 ExponentialDecay tuner
Input: μ_{init}, r
$\mu_{\text{ad}} = \mu_{init} \cdot c^r$
Return μ_{ad}

Algorithm 2 AdaptiveLoss tuner
Input: $\mu_{prev}, L_{\text{prev}}, L_{\text{curr}}$
$D = \frac{L_{\text{prev}} - L_{\text{curr}}}{L_{\text{prev}}}$
if D <= threshold:
if D <= 0:
$\mu_{\text{ad}} = \min(\mu_{\text{prev}} \cdot c_{\text{in}}, \mu_{max})$
else:
$\mu_{\text{ad}} = \max(\mu_{\text{prev}} \cdot c_{de}, \mu_{min})$
Return μ_{ad}

Apart from the two μ adjustment algorithms, the remaining part of the framework is the same as FedProx in [2]. The whole steps of the framework are shown in Algorithm 3.

Algorithm 3 FedProx with dynamic μ adjustment
Input: $\mu_{init}, R, K, N, E, p$
$\mu = \mu_{\text{init}}$
for $r = 0,1,\ldots,R-1$ do
Sever randomly choose a subset Sr devices from total K devices with the proportion p.
Sever sends the parameter of global model w^r to all chosen devices.
Each chosen device k updates its parameter w_k^{r+1} by using SGD to train local client model with objective function (3) for E epochs.
Each chosen device k sends its parameter to the sever.
Sever averages the w's according to their dataset proportion and get w^{r+1} for the global model.
If applying ExponentialDecay tuner
$\mu = \text{ExponentialDecay}(\mu_{\text{init}}, r)$
If applying AdaptiveLoss tuner
Test the global model and calculate its loss.
Update $\mu_{prev}, L_{\text{prev}}, L_{\text{curr}}$
$\mu = \text{AdaptiveLoss}(\mu_{\text{prev}}, L_{\text{prev}}, L_{\text{curr}})$
end for

Different from original FedProx framework, the author add a hyper-parameter μ adjustment part after the sever aggregating the client model parameters. Therefore, if the hyper-parameter μ adjustment part is ignored, the framework becomes to FedProx. Furthermore, if the initial μ is set to zero, the framework becomes to FedAvg.

4. Experiments

The paper first shows the experiment basic configuration in section 4.1. In section 4.2, 4.3, and 4.4, the paper presents the experiments and results on three different environment settings: (1) different initial μ (2) different data IID-degree (3) different client chosen fractions.

4.1. Basic configuration

The dataset used in this work is EMNIST that contains 62 classes of handwritten character digits. To achieve non-IID data on devices, the research applied Dirichlet distribution to partition the dataset by their classes, dividing it into training set and testing set by ratio of 4:1. Specifically, the parameter of Dirichlet controls the IID-degree of data. The larger the parameter value of Dirichlet, the more homogeneous the data. For the model utilized in this work, it is a simple neural network model composed of two convolutional layer and two fully connected layers. In addition, the optimizer used in training for all the cases is SGD, ensuring fairness in comparison with FedAvg. In client design, the total number of clients is 100. For chosen client in each round, it will train data with batch size 32, performing 5 local parameter updates in 5 local epochs with learning rate 0.1. For ExponentialDecay tuner, the multiplied constant used to decrease μ is 0.98. For AdaptiveLoss tuner, the multiplied constants used to increase and decrease μ are 1.02 and 0.98, the limitation range is from 0 to 1.

The evaluation metrics is based on the performance of global model, containing the loss value and prediction accuracy on test set with the number of training rounds. Besides, the baselines of this study are FedAvg (μ =0) and FedProx with static μ.

4.2. Experiments on different initial μ

In this section, the study fixed the Dirichlet parameter to 0.01 and client chosen fraction to 0.1, which means the data is extremely non-IID among clients and 10 clients will be randomly chosen in each round. The comparison between FedProx that applies two dynamic μ adjustment strategies and baselines with different initial μ is shown in Fig. 1. The initial μ values are set to [0.1, 0.05, 0.01, 0.005] to investigate the performance of frameworks under different μ. From top to bottom, Fig 1 presents the changes of loss value, accuracy, and μ value. Correspondingly, Fig 2 shows the comparison of average loss and accuracy in first t = [10, 20, 30, 100] round intervals and last 40 round interval.

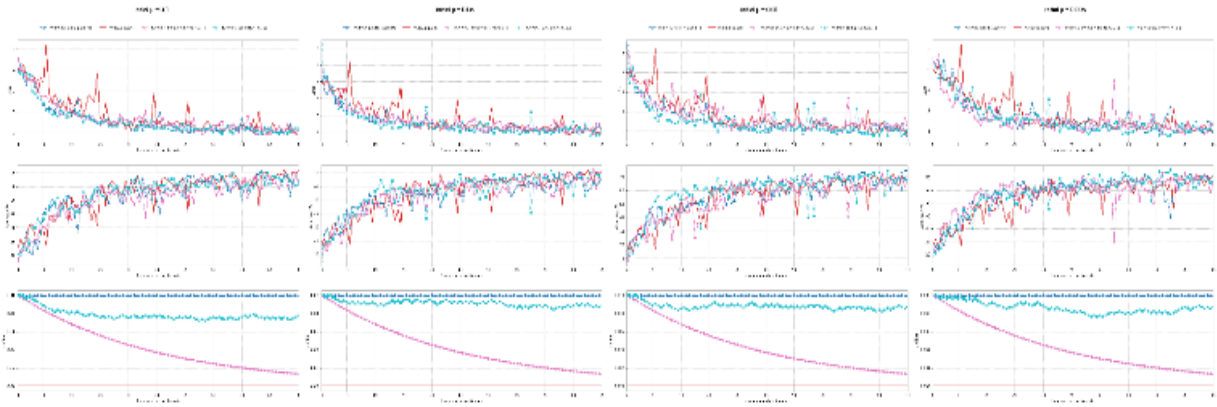


Fig. 1. Curve graph for the result of FedAvg and FedProx (static and dynamic μ) with different initial μ (Picture credit: Original).

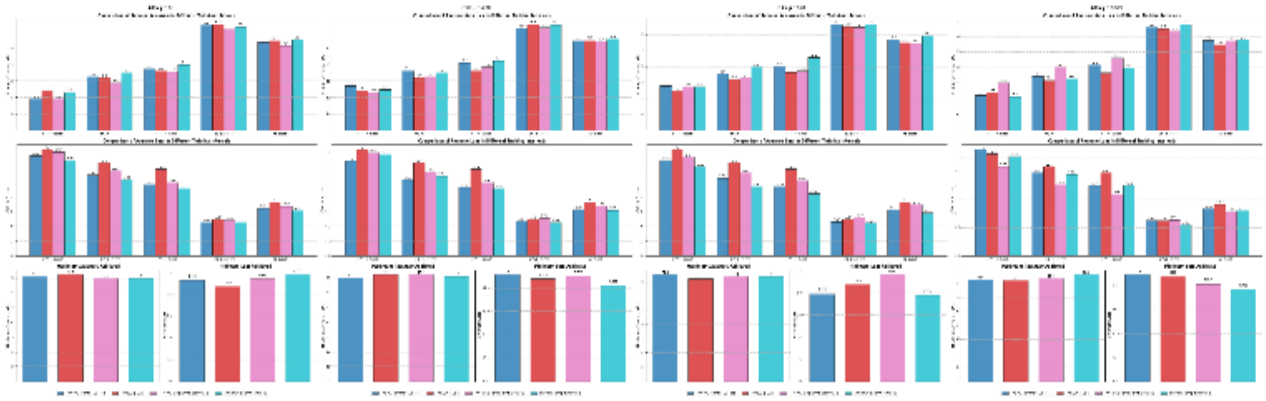


Fig. 2 Bar chart for average data and extremum comparison of FedAvg and FedProx (static and dynamic μ) with different initial μ (Picture credit: Original).

From Fig 1 and Fig 2, it is obvious that FedAvg has a relatively higher loss than FedProx (static and dynamic μ) in the early stages. By contrast, when initial μ = [0.1, 0.05, 0.01], FedProx with AdaptiveLoss tuner has the highest loss decrease speed in the early stages. Especially when initial μ =0.01, the average loss of FedProx with AdaptiveLoss tuner is 10.2% smaller than FedProx (static μ) in first 30 rounds. When initial μ =0.005, FedProx with ExponentialDecay tuner has the fastest loss decrease speed in the early stage and the average loss is 12.1% smaller than FedProx (static μ) in first 30 rounds.

In later stages of training, the average performance of FedProx with AdaptiveLoss tuner and initial μ = 0.005 in last 40 rounds is the best overall, where its average loss smaller 11.7% and average accuracy outperform 2.1% compared with FedProx (static μ).

On all training rounds, FedProx with AdaptiveLoss tuner and initial μ = 0.01 perform the best in average, where its average loss smaller 7.5% and average accuracy outperform 2.5% compared with FedProx (static μ). In extremum comparison, FedProx with ExponentialDecay tuner and initial μ = 0.005 achieve the lowest loss 0.769 (13.3% smaller than FedProx of static μ) and highest accuracy 76% (3.3% larger than FedProx of static μ). In a word, two dynamic μ adjustment strategies deed have some effects on original FedProx framework under some different initial μ values.

In terms of training stability, the loss and test accuracy of FedAvg fluctuate the most. As initial μ value decreases, training of FedProx(static and dynamic μ) becomes more unstable. This proves the function of stabilizing training in FL mentioned in [2].

4.3. Experiments with Different iid-Degrees

The research fixed the initial μ value to 0.01 and client chosen fraction to 0.1 in this section and IID-degrees of data (parameter value of Dirichlet) are set to [0.01, 0.1, 1]. In Fig 3 and Fig 4, the impact of IID-degree of data is very clear. As the IID-degree of data reduces, the overall loss decrease speed becomes slower and final accuracy also becomes smaller. In addition, when IID-degree is 0.1 and 1, there is no large performance difference between FedAvg and FedProx (static and dynamic μ). However, when IID-degree is 0.01, FedProx(static and dynamic μ) show their advantages of robustness to heterogeneous data. Furthermore, FedProx with AdaptiveLoss tuner performs the best and displays its strength in accelerating training process and reduce the minimum loss value compared to FedProx with static μ.

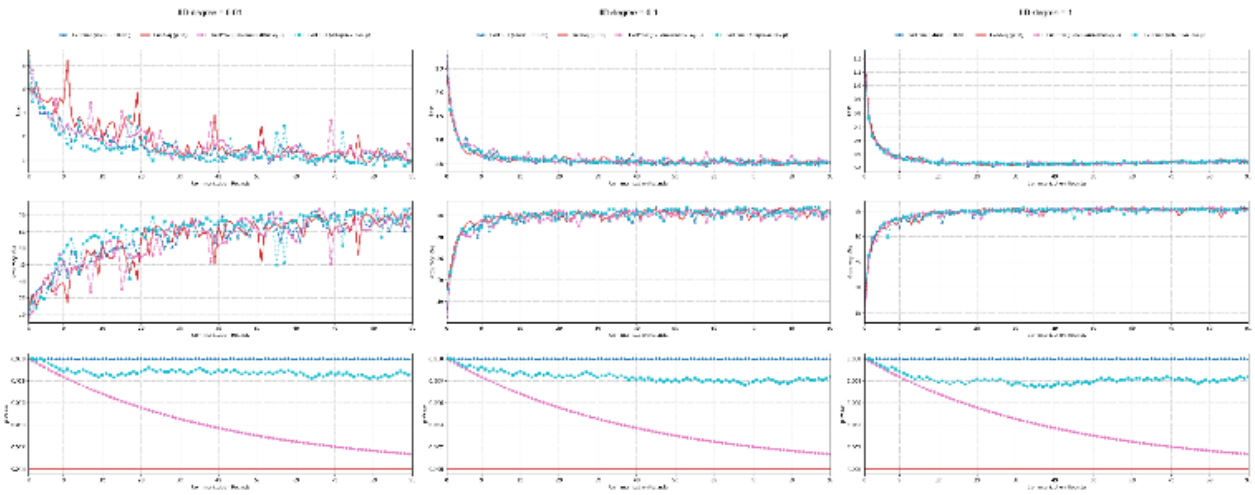


Fig 3. Curve graph for the result of FedAvg and FedProx (static and dynamic μ) with different IID-degree (Picture credit: Original).

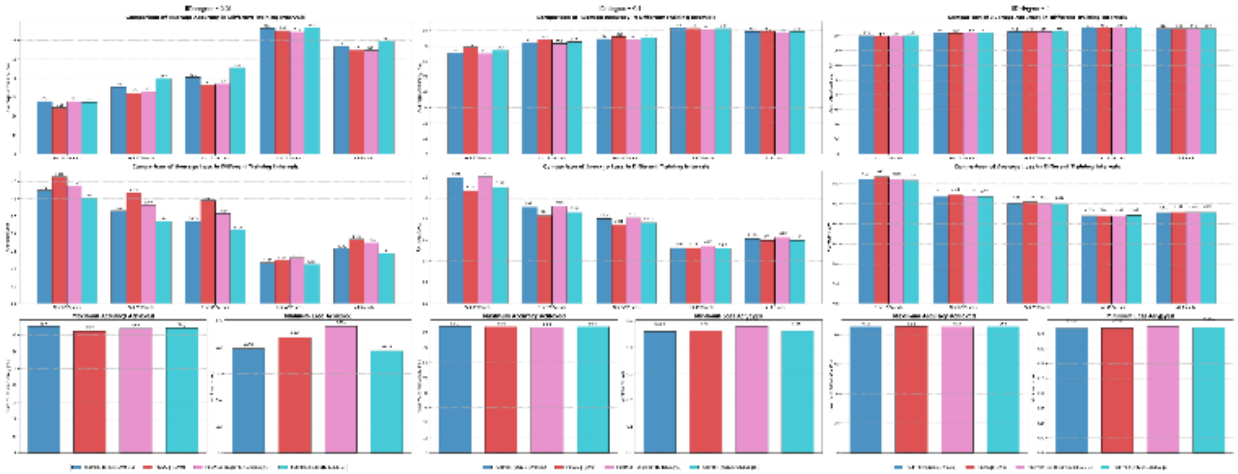


Fig 4. Bar chart for average data and extremum comparison of FedAvg and FedProx (static and dynamic μ) with different IID-degree (Picture credit: Original).

4.4. Experiments with Different Client Fractions

The IID-degree and initial μ are both fixed to 0.01 in this section and client fractions are set to [0.1, 0.5, 0.9]. From Fig 5 and Fig 6, it is obvious that higher client fraction could make the traning more stable and achieve better performance. When client fraction is 0.1, FedProx with AdaptiveLoss tuner performs the best (discussed before). When client fraction is 0.5, there is no big difference between FedAvg and FedProx (static and dynamic μ). When client fraction is 0.9, FedProx with ExponentialDecay tuner performs the best and presents high improvements compared to FedProx with static μ. Firstly, the average loss in first 30 rounds of FedProx with ExponentialDecay tuner is 8.3% lower than FedProx with static μ, indicating its faster training speed in early stage. Also, FedProx with ExponentialDecay tuner achieve the lowest loss 0.579 (7.1% smaller than FedProx of static μ) and highest accuracy 80.3% (0.8% larger than FedProx of static μ).

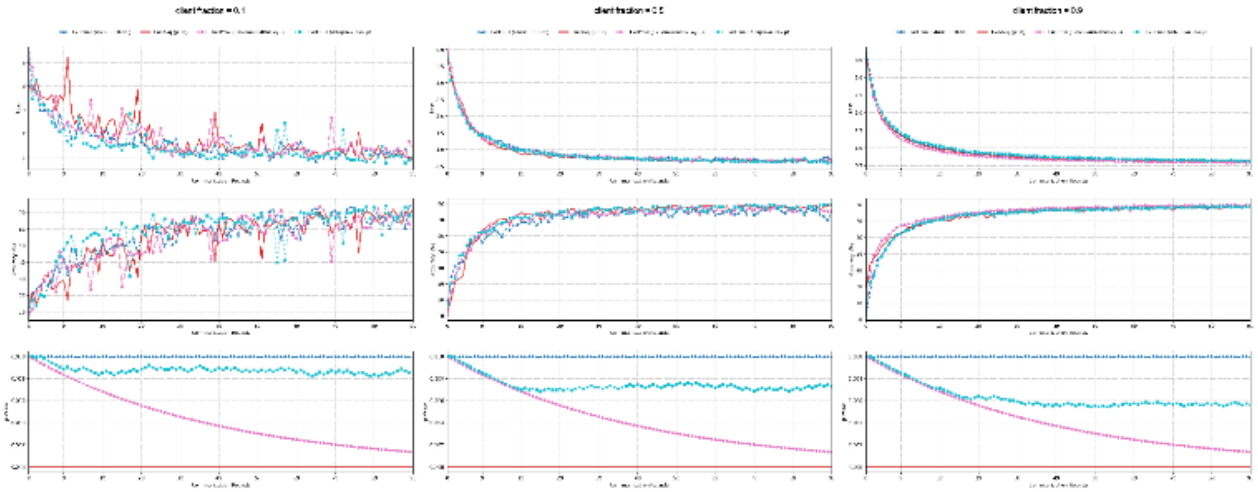


Fig 5. Curve graph for the result of FedAvg and FedProx (static and dynamic μ) with different client fractions (Picture credit: Original).

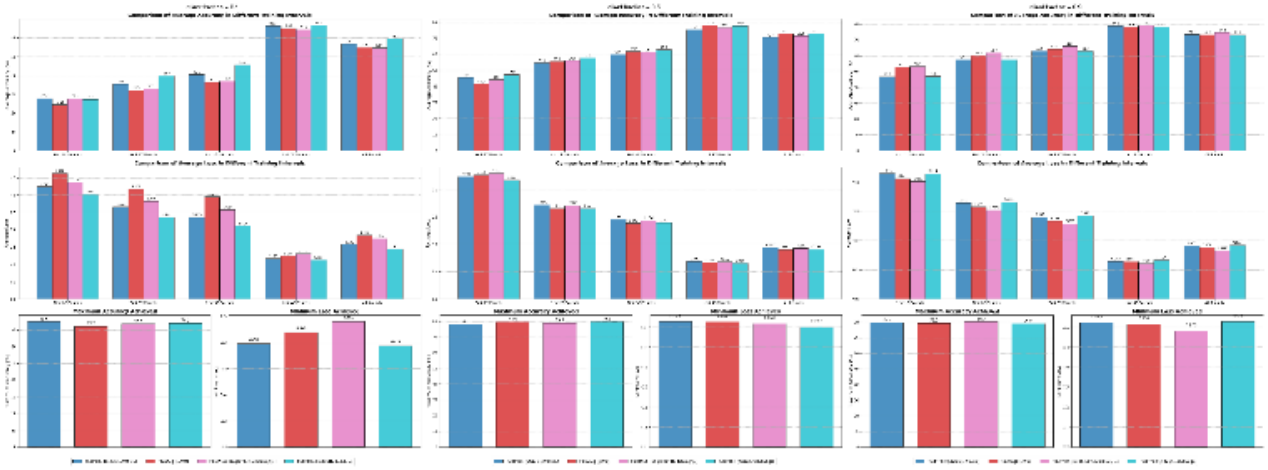


Fig 6. Bar chart for average data and extremum comparison of FedAvg and FedProx (static and dynamic μ) with different client fractions (Picture credit: Original).

5. Conclusion

This study proposes two dynamic hyper-parameter adjustment strategies for improving the performance of the FedProx algorithm in Federated Learning (FL), focusing on the adjustment of the hyper-parameter μ in the proximal term. Both strategies, ExponentialDecay and AdaptiveLoss, were designed to address the challenges caused by data heterogeneity among clients, which often leads to suboptimal performance in federated learning frameworks. The ExponentialDecay strategy gradually reduces μ by multiplying it with a constant factor, ensuring that client models converge efficiently without excessive drift. The AdaptiveLoss strategy adjusts μ based on the changes in loss values: it decreases μ when the loss decreases, and increases it when the loss increases, thus offering a more responsive approach to training dynamics.

The experiments conducted on the EMNIST dataset, which simulates real-world non-IID data distributions, demonstrated that both dynamic strategies outperform the conventional static μ setting. Specifically, the AdaptiveLoss strategy provided the best performance, reducing loss by 10.2% and improving accuracy by 2.5% compared to the static μ approach. These strategies were particularly effective in scenarios with highly heterogeneous data, leading to better model stability, reduced loss, and faster convergence. Additionally, the dynamic strategies were able to accelerate the training process, especially in the early stages, by adjusting μ based on the model's performance.

References

[1] McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Aguera y Arcas, B.: 'Communication-Efficient Learning of Deep Networks from Decentralized Data'. Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 2017, pp. 1–10 .

[2] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: 'Federated Optimization in Heterogeneous Networks'. Proceedings of Machine Learning and Systems (MLSys), 2020, 2, pp. 429–450

[3] Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: 'EMNIST: an extension of MNIST to handwritten letters'. Proc. 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, December 2017, pp. 1–6 .

[4] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: 'Federated Learning: Challenges, Methods, and Future Directions'. IEEE Signal Processing Magazine, 2020, 37, (3), pp. 50–60 .

[5] Tao, H., Zhang, J., Zhou, Y., et al.: 'Heterogeneous Federated Learning: State-of-the-art and Research Challenges'. ACM Computing Surveys, 2024, 56, (3), Article 79, pp. 1–44 .

[6] Wang, J., et al.: 'Knowledge Generation and Distillation for Road Segmentation in Intelligent Transportation Systems'. IEEE Transactions on Intelligent Transportation Systems, 2025.

[7] Zhu, X., et al.: 'RMER-DT: Robust Multimodal Emotion Recognition in Conversational Contexts Based on Diffusion and Transformers'. Information Fusion, 2025, 103268.

[8] Wang, R., et al.: 'RAFT: Robust Adversarial Fusion Transformer for Multimodal Sentiment Analysis'. Array, 2025, 100445.

[9] Chen, Y., Fang, F., Wang, B., et al.: 'An Efficient Federated Learning Framework for IoT: Intrusion Detection'. Proc. IEEE International Conference on Communications (ICC), Denver, CO, USA, October 2024 .

[10] Acar, D.A.E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P.N., Saligrama, V.: 'Federated Learning Based on Dynamic Regularization'. Proc. 9th International Conference on Learning Representations (ICLR), Virtual, 2021 .

[11] Haque, R.U., Markopoulos, P.: 'Federated Learning with Automated Dual-Level Hyperparameter Tuning'. IEEE Open Journal of Signal Processing, 2025, PP, (99), pp. 1–9 .