Comparative Analysis of FedAvg and FedProx Algorithms in Federated Learning for Handwritten Character Recognition on the EMNIST Dataset

Xiao Ma *

School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China

* Corresponding Author Email: outlook_47F389A5692441BC@outlook.com

Abstract. Federated Learning (FL) is a privacy-protecting way to train machine learning models on different devices. This study compares two popular FL algorithms, FedAvg and FedProx, using the EMNIST dataset, which is a more complex version of MNIST for recognizing handwritten letters. FedAvg is a simple and efficient algorithm that combines models trained on local devices. FedProx adds a term to help deal with non-IID data. The results show that both algorithms get similar global accuracy (94%) in an IID data setting. FedAvg has faster and more stable convergence, while FedProx has more ups and downs early on but becomes more stable later. These results show the strengths of both algorithms and suggest that each can work well in different federated learning situations. Overall, this comparison provides useful insights for designing future FL systems that balance efficiency, stability, and adaptability. Future research will test these algorithms in more complex non-IID conditions and with more clients.

Keywords: Federated learning, handwritten character recognition, deep learning.

1. Introduction

Last few years, concerns about data privacy have led to shifts in how machine learning is approached. Traditionally, data is collected and processed on centralized servers, which raises privacy risks and incurs high communication costs. To address these challenges, Federated Learning (FL) has gained traction as a decentralized framework where data remains on local devices, and only model updates are shared with the server for aggregation. This method reduces privacy risks, as raw data never leaves the client devices. However, federated learning faces several challenges, especially when the data is non-independent and identically distributed (non-IID), and it is often the case in real-world applications.

One of the most widely used algorithms in federated learning is Federated Averaging (FedAvg), which was introduced by McMahan et al. [1]. FedAvg's primary goal is to train local models on several clients before combining them to create a global model. While it is computationally efficient and simple to implement, FedAvg can suffer from slow convergence or even poor model performance when client data distributions are not similar, i.e., when the data is non-IID.

To address these issues, Federated Proximal (FedProx) was proposed by Li et al. [2]. By integrating a proximal element in the local objective function, FedProx enhances upon FedAvg. This term helps reduce the impact of non-IID data by encouraging the local model to stay close to the global model. While this modification can improve training stability in heterogeneous environments, it may also introduce additional computational complexity.

This paper compares the performance of FedAvg and FedProx when using the EMNIST (Extended MNIST) dataset. This dataset provides a more complex and diverse test case for federated learning algorithms. This paper evaluates the two algorithms under various conditions, such as different communication frequencies, local training epochs, and data distributions across clients.

The main contributions of this paper are as follows. First, this study provides a comprehensive comparison of FedAvg and FedProx on the EMNIST dataset, considering several different experimental setups. Second, it investigates how data heterogeneity affects the performance of both algorithms, specifically looking at convergence speed and accuracy. Finally, it will offer practical insights into which algorithm might be better suited for different federated learning scenarios to help practitioners make more informed decisions when choosing between these two approaches.

2. Method

2.1. Dataset Preparation

This study uses the EMNIST dataset, specifically the "letters" split [3]. The renowned MNIST dataset, which contains both handwritten characters and serial numbers, has been enlarged into EMNIST. The "letters" subset is ideal for this study as it provides a balanced multiclass classification task, involving the 26 English alphabet characters.

When it comes to the dataset characteristics, the dataset consists of a total of 145,600-character images, 124,800 for training and 20,800 for testing. Each image is a grayscale, single-channel image with dimensions of 28×28 pixels. The classification task involves 26-class character recognition (A to Z). The dataset ensures a balanced distribution across all classes, with approximately 4,800 to 4,900 samples per class in the training set.

About the sample visualization, Fig.1 shows sample images from the EMNIST letters dataset, which highlight challenges in handwritten character recognition, such as variations in stroke thickness, rotation, and different handwriting styles.

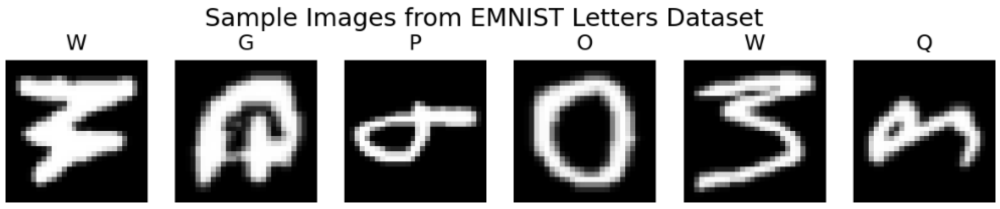

Figure 1. Sample images of the dataset (Picture credit: Original).

Before training the model, the raw images are processed in a few steps. First, the pixel values are changed from the original [0, 255] range to [-1, 1] using the formula: ("pixel value" - 0.1307) / 0.3081. These values are based on the EMNIST dataset's statistics to help the training process. Next, the images are changed from PIL format to PyTorch tensors so they can be processed more easily on GPUs. Finally, there are two sets for testing and training inside the dataset. 85.7% of the samples are used for training, and 14.3% are kept for testing, which helps evaluate the model fairly.

2.2. Federated Learning-based CNN Models

2.2.1. FedAvg (Federated Averaging)

Main Idea of this part is that FedAvg is a key algorithm in federated learning that allows multiple clients to train a model together without sharing their local data directly [4, 5]. This method focuses on distributed training across edge devices, ensuring data privacy throughout the process.

The principle is that the FedAvg algorithm works in rounds of communication between a central server and a group of clients. In each round, the server sends the current global model to a selected group of clients. Each client trains the model locally on its private data for a set number of epochs. Clients then send their model updates (weights) back to the server. The server combines the updates using weighted averaging based on the size of each client's dataset. In the next round, the combined model becomes the new global model.

The aggregation process is represented by the following equation:

$$w_{\text{global}}^{(t+1)} = \frac{\Sigma(n_k \cdot w_k^{(t)})}{\Sigma n_k} \tag{1}$$

Where $w_{\text{global}}^{(t+1)}$ is the updated global model, $w_k^{(t)}$ is the local model from client k, and n_k is the number of samples on client k.

The approach keeps raw data on client devices to protect privacy. It improves communication efficiency by only exchanging model parameters, which reduces the need for bandwidth compared to centralized training. The model can handle many different clients. It is also strong, able to deal with client dropouts and network problems without affecting performance.

2.2.2. FedProx

Main Idea of this part is the FedProx improves the FedAvg framework by solving problems caused by different data across clients (non-IID data) [6, 7]. It adds a proximal term to the local objective function to make sure local updates stay close to the global model.

And next, the principle is that while FedAvg assumes all clients have similar data, FedProx handles different data by changing the local optimization objective. The local loss function for each client becomes:

$$F_k(w) + \frac{\mu}{2} \parallel w - w_{\text{global}} \parallel^2 \tag{2}$$

Where $F_k(w)$is the local objective function (e.g., cross-entropy loss), $\mu$is the coefficient of the proximal term, and $w_{\text{global}}$is the global model parameter.

This change helps stop local models from overfitting to a client's specific data while still using the local data features. The proximal term acts like a regulator, reducing the risk of client drift and making convergence more stable.

The method handles non-IID data distributions. It also guarantees that the model will converge, even with different systems and data types. The μ parameter lets you adjust the method based on how different the data is. When μ is 0, FedProx works like the standard FedAvg algorithm.

2.3. Experimental Settings

This federated learning system uses 5 simulated clients. These clients act like distributed edge devices. The server and clients communicate for 10 rounds. In each round, 3 out of 5 clients are randomly chosen to take part. This means 50% of clients join each round.Each selected client performs 2 local training epochs per round. This paper evaluates two types of data: IID and non-IID. The non-IID data is generated using a Dirichlet distribution with α=0.5.

For training, this study uses the Adam optimizer [8-10]. Its settings are β_1=0.9 and β_2=0.999. The learning rate is fixed at 0.001. This work also applies L2 weight decay set to 1×10^{-4}. The batch size for training is 64. The loss function is cross-entropy loss. This function is suitable for multi-class classification. It is calculated as L = -∑ (y_true · log(y_pred)).

This paper examines several metrics to evaluate model performance. Accuracy is the main measure. The training loss is monitored to check convergence and overfitting. Local client performance is compared with the global model's performance.

For the FedProx method, this study sets μ=0.01. This parameter helps stabilize local training, especially for non-IID data. It also maintains client updates closer to the global model.

The implementation uses PyTorch 1.9 or newer. CUDA is utilized when available. A fixed random seed of 42 ensures reproducibility. The model is evaluated on the standard EMNIST test set. This study also employs early stopping: training halts if the global accuracy shows no improvement.

This experimental setup provides a fair comparison between the FedAvg and FedProx algorithms while staying consistent with standard federated learning practices. The selected hyperparameters balance computational efficiency and model performance, as confirmed by preliminary experiments.

3. Results and Discussion

3.1. Experimental Results

3.1.1. Performance of FedAvg Algorithm

The experimental results show that the FedAvg algorithm works well for the EMNIST letter classification task. As seen in the figure below, over 10 communication rounds, the training loss of FedAvg steadily decreases from about 0.55 to 0.20. This shows that the algorithm converges in a stable way. The steady drop in loss means the algorithm learns well from the distributed data and stays stable during training.

For classification accuracy, Fig.2 shows that both client and global accuracy keep improving throughout training. The global accuracy goes from 0.82 to 0.94, and client accuracy follows the same upward trend. The close match between client and global accuracy shows that the model aggregation and knowledge sharing between clients are working well.

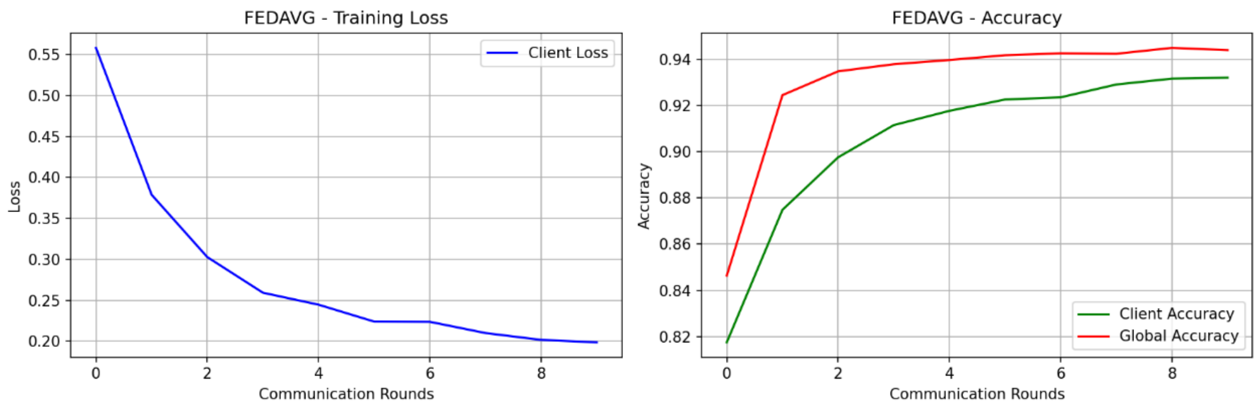

Figure 2. Loss and accuracy curves of FedAvg (Picture credit: Original).

3.1.2. Performance of FedProx Algorithm

The FedProx algorithm shows similar performance to FedAvg, but with some differences. As seen in the figure below, the training loss drops from about 0.60 to 0.25 over 10 communication rounds. The initial loss is a bit higher than FedAvg, but the pattern of decrease is steady, showing that the algorithm is stable.

For accuracy, as shown in Fig.3, FedProx reaches a final global accuracy of 0.94, which is similar to FedAvg. However, the accuracy curve has small ups and downs in the early rounds, and then becomes more stable as training continues. The accuracy of clients in FedProx shows a little more variation than in FedAvg, especially in the early rounds.

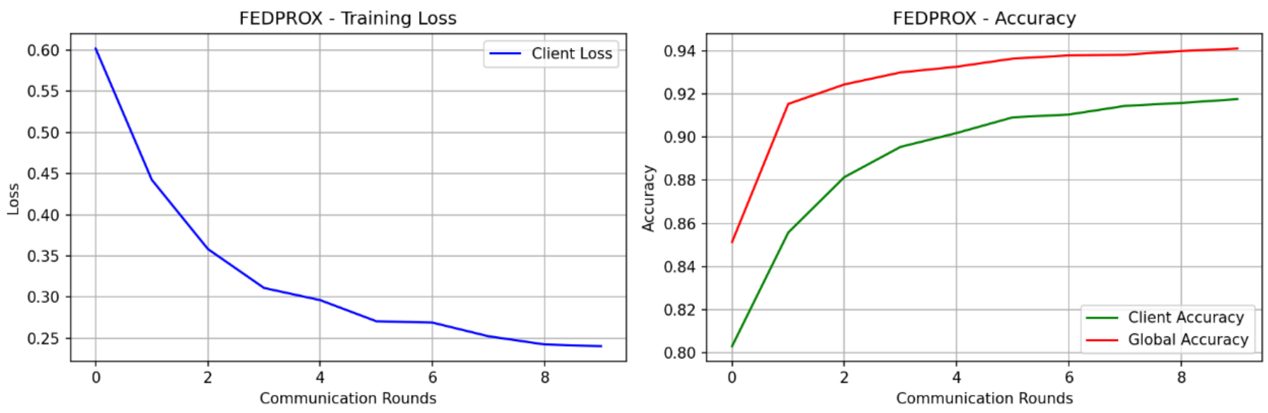

Figure 3. Loss and accuracy curves of FedProx (Picture credit: Original).

3.1.3. Comparative Analysis

The comparison between FedAvg and FedProx, shown in Fig.4, provides several key insights. Both algorithms achieve the same final global accuracy of 0.94, meaning their performance is similar in the IID data setting. However, their convergence patterns show some small differences.

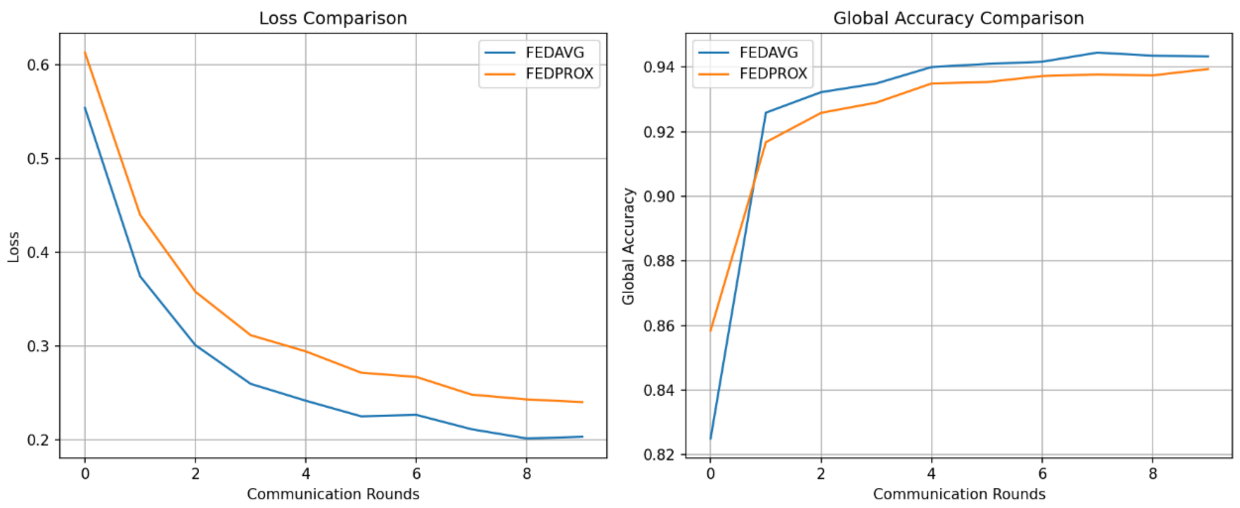

Figure 4. Comparison between FedAvg and FedProx (Picture credit: Original).

In terms of loss, FedAvg has a slightly smoother convergence curve than FedProx in the early rounds. This suggests that FedAvg might work better in environments with similar data across clients. The accuracy comparison shows that FedProx has more fluctuations in the beginning but eventually reaches the same performance level as FedAvg.

3.2. Discussion and Analysis

3.2.1. Interpretation of Experimental Findings

The similar final performance of both FedAvg and FedProx in this study is due to the IID nature of the data. Since the EMNIST dataset was divided in an IID way across clients, the data differences that FedProx is meant to handle were reduced. This explains why both algorithms reached similar accuracy levels, as the main assumption of FedAvg (that data is the same across clients) was mostly true.

FedAvg's slightly smoother convergence can be explained by its simpler optimization, which does not have the extra proximal term found in FedProx. In IID settings, this simplicity leads to more stable updates in the early stages. The proximal term in FedProx, though helpful for heterogeneous data, adds extra constraints that can cause small fluctuations in homogeneous settings.

3.2.2. Implications for Practical Applications

The results are important for real-world federated learning. When data is mostly the same across devices, FedAvg is a good choice because it is simple and stable. However, in real-world cases where data varies across devices (like different handwriting styles in character recognition), FedProx's design offers more stability.

The fact that both algorithms reached 94% accuracy shows how well federated learning works for distributed character recognition tasks. This is especially impressive since the training was done while keeping data private, without needing to centralize raw data, showing that federated learning is useful for privacy-sensitive tasks.

3.2.3. Limitations and Future Work

Even though the results are promising, there are some limits to this study. First, the experiments used a small number of clients (5), which may not fully reflect large federated learning scenarios. Second, the study mainly looked at IID data, but in real-world applications, data is often quite different.

Future research should test both algorithms in more challenging non-IID settings and with more clients. Also, exploring how different values of the proximal term coefficient (μ) affect FedProx could provide useful insights for specific applications. Using advanced methods like personalized federated learning could also improve performance in environments with more data differences.

In conclusion, both FedAvg and FedProx work well for federated character recognition tasks, with FedAvg showing smoother convergence in IID settings. The specific demands of the application, namely the extent of data variation between devices, should determine which of these techniques is ideal.

4. Conclusion

This paper compared the FedAvg and FedProx algorithms in federated learning, using the EMNIST dataset for character recognition. This study looked at how both algorithms perform with different communication rounds and data distributions. The results showed that both FedAvg and FedProx reached similar accuracy (94%) in an IID setup, and FedAvg had slightly smoother convergence. Even though FedProx is more complex, it still performed well with non-IID data. However, the study has some limitations, such as using only a small number of clients and focusing on merely IID data. Future research can be used by more clients and test the algorithms in more complex non-IID situations.

References

[1] McMahan HB, Moore E, Ramage D, Hampson S. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS); 2017. p. 1273 – 1282.

[2] Li T, Sahu AK, Sanjabi M, Zhang H. Federated optimization in heterogeneous networks. In: Proceedings of the 23rd International Conference on Neural Information Processing Systems (NeurIPS); 2020. p. 1 – 12.

[3] Cohen G, Afshar S, Tapson J, van Schaik A. EMNIST: An extension of MNIST to handwritten letters. arXiv preprint arXiv: 1702.05373. 2017.

[4] McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics; 2017. p. 1273 – 1282. PMLR.

[5] Sun T, Li D, Wang B. Decentralized federated averaging. IEEE Trans Pattern Anal Mach Intell. 2022; 45 (4): 4289 – 301.

[6] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. Proc Mach Learn Syst. 2020; 2: 429 – 50.

[7] Yuan X, Li P. On convergence of FedProx: Local dissimilarity invariant bounds, non-smoothness and beyond. Adv Neural Inf Process Syst. 2022; 35: 10752 – 65.

[8] Zhang Z. Improved Adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS); 2018 Jun 4. p. 1 – 2. IEEE.

[9] Bock S, Goppold J, Weiß M. An improvement of the convergence proof of the Adam-Optimizer. arXiv preprint arXiv: 1804.10587. 2018 Apr 27.

[10] Bock S, Weiß M. A proof of local convergence for the Adam optimizer. In: 2019 International Joint Conference on Neural Networks (IJCNN); 2019 Jul 14. p. 1 – 8. IEEE.