

Performance Comparison and Analysis Between MOON and FedProx in Image Classification

Xuanzhou Shi^a

Miami College of Henan University, Henan University, Kaifeng, China

Keywords: Federated Learning, Data Heterogeneity, Image Classification.

Abstract: Traditional machine learning algorithms mostly follow a centralized training paradigm, which poses challenges to data security and privacy, and restricts the in-depth application of artificial intelligence models in many fields. As a great way to train models in siloed data, federated learning has emerged in recent years and attracted much attention from the industry and academic community. Though much effort has been devoted, challenges persist within federated learning frameworks when addressing data that is identically distributed yet lacks independence. This paper focuses on introducing the latest research to solve this problem and quantitatively discusses their performance on various datasets, aiming to provide the decision-making basis for algorithm selection. Specifically, three representative federated learning algorithms are first introduced, including their design ideas and key steps. This paper further analyzes the advantages and disadvantages of these methods by comparing the performance in accuracy, communication efficiency, different numbers of local epochs, and different heterogeneous environments. Extensive results show that MOON is superior to FedProx in all aspects of the experiment, showing the superiority of MOON in image classification.


1 INTRODUCTION

The huge success of AlphaGo has raised great expectations for this kind of big data-driven AI to be realized in various industries. However, the reality is that in addition to a few industries, many industries do not have sufficient data and good quality data to support the implementation of artificial intelligence. At the same time, due to the restrictions of laws and regulations, data often appear in the form of islands, and the data of all parties cannot be transferred to a centralized server to train the model. In this context, federated learning (Li et al., 2020) has been gaining traction as a viable option for addressing privacy concerns and ensuring data security, garnering increasing interest in recent times.

Federated learning can keep the original data in the local client. During every training cycling, participants are required only to transmit their updated models to a central server, which then amalgamates these local models to formulate a comprehensive global model. In this process, the original data are not exchanged, which solves the above problems well, which attracts the research

interest of many people and obtains a wide range of research and applications (Zhang et al., 2022; Yang et al., 2022), including medical image assessment, processing of natural language, auto driving. However, the heterogeneity of data distribution (Zhao et al., 2018) is still an open issue that hinders the further development of federated learning. Data between participants may have different feature distributions, data types, or data sizes. This heterogeneity leads to potential inefficiencies in federated learning, particularly during the model training and amalgamation process, where significant disparities can emerge between the average global model as well as the global optimum, especially when local models are updated and their specific objectives diverge substantially from the overarching global objective.

To solve the data heterogeneity problem, methods such as feature alignment, data transformation, cross-device transfer learning, and aggregation strategy adjustment can be adopted. MOON (Li et al., 2021) and FedProx (McMahan et al., 2017) are two representative algorithms. Model-Contrastive Federated Learning (MOON) introduces model

^a <https://orcid.org/0009-0005-8574-7262>

contrastive loss during local training to correct update direction. The objective of comparing model losses is to enhance the distinction between features generated in the current model iteration and those from preceding iterations, while simultaneously reducing the discrepancies involving features produced via the global and updated models. This is because on the server side, the global model has a tendency to produce better features in comparison to models that are updated locally. FedProx allows local models with inadequate training to handle system heterogeneity issues. Regardless of whether a local model has finished its training (fixed Epoch), FedProx combines all participating local models, meaning that it does not necessitate all local models to be trained with precision. Concurrently, FedProx addresses the issue of frequent local model updates hindering the global model's convergence by integrating a proximal term to the local model's loss function. This addition acts as a deterrent against excessive deviations of local models from the global models.

Aiming to provide decision-making basis for federated learning algorithms selection, this paper compares the performance of MOON and FedProx on image datasets of deep learning models, and finds the superiority of MOON over FedProx in image classification. Specifically, in Section 2, three representative federated learning algorithms are first introduced, including their design ideas and key steps. Section 3 compares the performance of different methods on three datasets. Section 4 concludes this work and discusses the future development of federated learning.

2 METHOD

This section first revisits the classic federated learning algorithm FedAvg (Li et al., 2020) and gives the problem definition of data heterogeneity. Then, MOON and FedProx are detailed in the following section.

2.1 Problem Definition

FedAvg (Li et al., 2020) is a strategy that utilizes Stochastic gradient descent (SGD) for updates, which performs effectively in real life, and the steps of every round are shown in Figure 1. Initially, the server selects clients for federated learning participation and dispatches the model to them. Subsequently, clients utilize Stochastic Gradient Descent (SGD) for model refinement using their local datasets. Afterward,

clients transmit the refined model back to the central server. In the final step, the central server amalgamates and evaluates the models, with the evaluation outcomes serving as a benchmark for subsequent training rounds.

However, FedAvg is not suitable for heterogeneous environments (Li et al., 2019), where the data distribution on different devices may be quite different. Some devices may have more or fewer samples, or there may be differences in data quality. This leads to some devices contributing inconsistently to the federated learning process, which affects global performance. To solve this problem, the current studies can be primarily categorized into two groups: improving the local training phase and improving the aggregation phase (corresponding to the first and fourth steps of Figure 1, respectively). The two models in this paper belong to the first category.

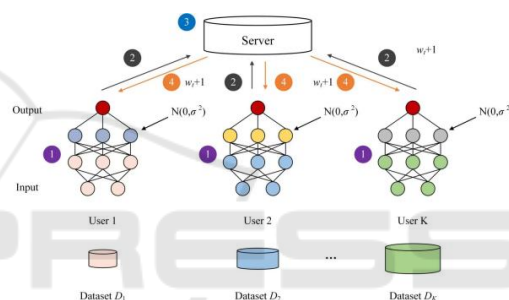


Figure 1: The workflow of FedAvg algorithm.

2.2 MOON

Contrastive learning (Jaiswal et al., 2020) is a machine learning method that aims to learn useful feature representations by comparing the similarities and differences between different samples, and has made positive contributions to learning visual representations. The main concept is that comparable samples are assigned to a corresponding embedding space, ensuring that similar samples are in closer proximity within the embedding space while dissimilar samples are more distant from every other. Inspired by the concept of contrastive learning, MOON was developed as an adaptation of FedAvg with modifications to local training. MOON is designed to narrow the divergence between representations acquired by the local model and those by the global model, enhancing the learning efficacy of the global model (Li et al., 2021). Concurrently, it amplifies the disparity involving the current local model's representations and those from its preceding iterations, mitigating the effects of drift during local training in order to accommodate accuracy in diverse environments.

The architecture of MOON's network comprises three components: an output layer, a projection head, and a foundational encoder, as depicted in Figure 2. The foundational encoder extracts vectors from the input. Through the projection head, inputs are transformed into a fixed-dimensional space. The output layer is responsible for generating the predicted values for every class. At a weight of w , the function $F_w(\cdot)$ represents the entire network, while $G_w(\cdot)$ represents the network excluding the output layer.

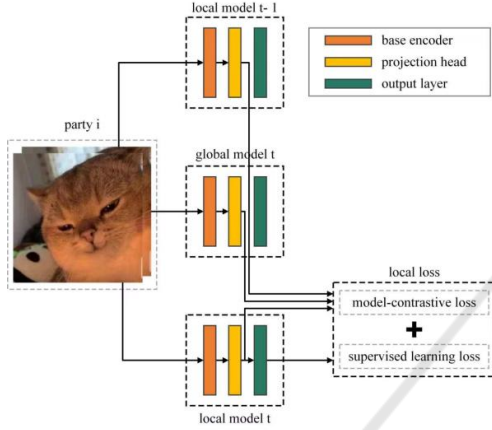


Figure 2: The workflow of FedAvg algorithm.

The training procedure of MOON is shown in Figure 3. The MOON has three representations during the local training phase: (1) the output derived from the most recent model trained locally and then transmitted to the server ($Z_{prev} = R_{w_i^{t-1}}(x)$), (2) the global model sends the representation to the local at the start of every round ($Z_{glob} = R_{w^t}(x)$), and (3) the output derived from the current update of the local model ($Z = R_{w_i^t}(x)$). The training objective is to have z close to Z_{glob} and z far away from Z_{prev} .

The local loss in training is divided into two parts: the initial component is referred to as the cross-entropy loss, labeled as ℓ_{sup} , while the subsequent component represents the characteristic ℓ_{con} in MOON. ℓ_{con} is defined as follows:

$$\ell_{con} = -\log \frac{a}{a+b} \quad (1)$$

$$a = \exp(\text{sim}(z, z_{glob})/\tau) \quad (2)$$

$$b = \exp(\text{sim}(z, z_{prev})/\tau) \quad (3)$$

Where, $\text{sim}(\cdot)$ denotes the cosine similarity function, and τ represents a temperature-related parameter. The optimization objectives of MOON are expressed as:

$$\ell = \ell_{sup}(w_i^t; (x, y)) + \mu \ell_{con}(w_i^t; w_i^{t-1}; w^t; x) \quad (4)$$

2.3 FedProx

2.3.1 Optimization Points for FedProx

Federated learning frameworks engage multiple clients and a central server in collaboration, aiming to minimize:

$$\min_w f(w) = \sum_{k=1}^N p_k F_k(w) \quad (5)$$

The objective function $f(w)$ is used to describe the objective or loss of the optimization problem. N is the number of participating devices. p_k is the weight factor such that $p_k \geq 0$ and the sum of all weight factors is one. $F_k(w)$ symbolizes the local objective function, which quantifies the local empirical risk for different data distributions D_k .

One common approach in federated optimization to minimize communication is to employ a local objective function derived from device data, as opposed to utilizing a global objective function, which is proxied on the client side. During every outer iteration, a subset of devices is chosen to optimize a local objective function using a local solver. Subsequently, device-generated updates are relayed to the central server, where they undergo amalgamation to refine the global model. The crucial factor for achieving flexible performance in this situation is the ability to accurately solve every local objective, allowing for the calibration of local computation against communication, contingent on the volume of local iterations completed. For this reason, this definition is formally proposed below, which will be used continuously in the following papers.

Definition 1 (γ -inexact solution). For a function $h(w; w_0) = F(w) + \frac{\mu}{2} \|w - w_0\|^2$, and $\gamma \ni [0, 1]$, w^* represents a γ -inexact solution of $\min_w h(w; w_0)$ if $\|\nabla h(w^*; w_0)\| \leq \gamma \|\nabla h(w_0; w_0)\|$, where $\nabla h(w; w_0) = \nabla F(w) + \mu(w - w_0)$. Note that a smaller γ corresponds to higher accuracy.

It is crucial to optimize the tuning of hyper-parameters in FedAvg, especially the number of epochs for local training plays an important role in convergence, the details are in Algorithm 2 in Figure 4. Executing more local training times can reduce the number of communications, but in a heterogeneous environment, more local updates may affect the convergence, even deviate from the global goal, and may make the device fail to complete the number of updates on time. With the change of local data and available system resources, the number of local updates will be different, so it is necessary to increase the convergence robustness and increase the number of local updates as much as possible. Obviously,

FedAvg does not meet these conditions, so it is proposed that FedProx can solve this problem.

```

Algorithm 1 The framework of MOON


---


Input: the total amount of communication rounds, denoted as  $T$ 
        number of parties  $N$ , the quantity of local Epochs  $E$ ,
        temperature  $\tau$ , learning rate  $\eta$ , hyper-parameter  $\mu$ 


---


Output: The final model  $w^T$ 
Server executes:
initialize  $w^0$ 
for  $t = 0, 1, \dots, T-1$  do
    for  $i = 1, 2, \dots, N$  in parallel do
        send the global model  $w^t$  to  $P_i$ 
         $w_i^t \leftarrow \text{PartyLocalTraining}(i, w^t)$ 
         $w^{t+1} \leftarrow \sum_{k=1}^N \frac{|D^k|}{|D|} w_k^t$ 
    return  $w^T$ 
PartyLocalTraining( $i, w^t$ ):
 $w_i^t \leftarrow w^t$ 
for epoch  $i = 1, 2, \dots, E$  do
    for each batch  $b = \{x, y\}$  of  $D^i$  do
         $\ell_{sup} \leftarrow \text{CrossEntropyLoss}(F_{w_i^t}(x), y)$ 
         $z \leftarrow R_{w_i^t}(x)$ 
         $z_{glob} \leftarrow R_{w^t}(x)$ 
         $z_{prev} \leftarrow R_{w_i^{t-1}}(x)$ 
         $\ell_{con} \leftarrow -\text{Log} \frac{\exp(\text{stm}(z, z_{glob})/\tau)}{\exp(\text{stm}(z, z_{glob})/\tau) + \exp(\text{stm}(z, z_{prev})/\tau)}$ 
         $\ell \leftarrow \ell_{sup} + \mu \ell_{con}$ 
         $w_i^t \leftarrow w_i^t - \eta \nabla \ell$ 
    return  $w_i^t$  to server
    
```

Figure 3: The training procedure of MOON algorithm.

```

Algorithm 2 Federate Averaging. The  $K$  clients are index by  $k$ ;  $B$  is the local minibatch
size,  $E$  is the number of local epochs, and  $\eta$  is the learning rate.


---


Server executes:
initialize  $w^0$ 
for each round  $t = 1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow$  (A set of  $m$  clients selected at random)
    for each client  $k \in S_t$  in parallel do
         $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$ 
     $m_t \leftarrow \sum_{k \in S_t} n_k$ 
     $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{m_k}{m_t} w_{t+1}^k$ 
ClientUpdate( $k, w$ ):
     $\beta \leftarrow$  (split  $P_k$  into batches of size  $B$ )
    for epoch  $i = 1, 2, \dots, E$  do
        for batch  $b \in \beta$  do
             $w \leftarrow w - \eta \nabla \ell(w; b)$ 
    return  $w$  to server
    
```

Figure 4: The training procedure of federate averaging.

2.3.2 FedProx

The framework of FedProx is similar to that of FedAvg in that, in every round, devices are selected, global models are sent, local updates are performed, and then these models are aggregated. The distinction lies in the fact that FedProx enables the enhancement of FedAvg by varying the workload carried out locally on devices based on system resources, and then combining incomplete solutions submitted by slow performers. FedProx allows local devices to perform a variable amount of work according to their available system resources, instead of dropping backward devices, that is, different devices can perform different iterations, provide variable γ , and extend from Definition 1 to Definition 2.

Definition 2 (γ_k^t -inexact solution). For a function $h_k(w; w_t) = F_k(w) + \frac{\mu}{2} \|w - w_t\|^2$, and $\gamma \in [0, 1]$, w^* is a γ_k^t -inexact solution of $\min_w h_k(w; w_t)$ if $\|\nabla h_k(w^*; w)\| \leq \gamma_k^t \|\nabla h_k(w_t; w_t)\|$, where $\nabla h_k(w; w_t) = \nabla F_k(w) + \mu(w - w_t)$. Note: a smaller γ_k^t corresponds to higher accuracy.

It can be seen that from Definition 1 to Definition 2, the objective function and gradient calculation method are mainly changed to local functions on every device to adapt to the heterogeneous environment. The γ_k^t approximate solution defined in this paper can measure the precision of the solution on every device, and evaluate the overall optimization more comprehensively.

After adjusting to the diverse environment, it is important to be aware that excessive local updates may lead to divergence in the approach as a result of the varying underlying data. To solve this problem, the above effects can be effectively avoided by adding the proximal term. Rather than directly minimizing a local function $F_k(\cdot)$ expressed below, Device K employs its selected local solver to approximate the minimization of the intended target.

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} \|w - w^t\|^2 \quad (6)$$

The training procedure of FedProx is shown in Algorithm 3 in Figure 5. In particular, the FedAvg can be seen as a specific instance of FedProx, where $\mu = 0$, the specially chosen local solver is SGD, and the constant γ between devices is fixed.

Algorithm 3 FedProx(Proposed Framework)

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \dots, N$

for $t = 0, \dots, T - 1$ **do**

The server randomly chooses a subset S_t from K devices

(each device k is chosen with a likelihood of p_k)

The server transmits w^t to all selected devices.

Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t -inexact

minimizer of: $w_k^{t+1} \approx \arg \min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} \|w - w^t\|^2$

Each device $k \in S_t$ sends w_k^{t+1} back to the server

Server aggregates the w 's as $w^{t+1} = \frac{1}{K} \sum_{k \in S_t} w_k^{t+1}$

end for

Figure 5: The training procedure of federate averaging.

3 EXPERIMENT

3.1 Original Datasets

This study undertakes a comparative analysis of MOON, FedProx, and FedAvg across CIFAR-10, CIFAR-100, and Tiny-Imagenet datasets (<https://www.kaggle.com/c/tiny-imagenet>). Moreover, it employs two distinct network architectures: ResNet-50 (He et al., 2016) as the primary encoder for CIFAR-100 and Tiny-Imagenet, and a convolutional neural network (CNN) for CIFAR-10. The CNN consists of 2 fully connected layers with ReLU activation (the first layer having 120 units and the second 84 units), along with five convolutional layers and a pooling layer (the initial layer with 6 channels, followed by one with 16 channels). A two-layer MLP serves as the projection head for all datasets, with the output dimension set to 256 by default.

3.2 Experimental Settings

PyTorch is used to implement MOON, FedProx and FedAvg. SGD optimizer is adopted for all three methods, where the learning rate is established at 0.01, accompanied by a weight decay of 0.00001 and momentum of 0.9. The batch size is determined as 64. For the CIFAR-10, CIFAR-100, and Tiny-Imagenet datasets, the amount of communication rounds is designated as 100, 100, and 20, respectively. For MOON, the default temperature parameter is 0.5. In the article, Dirichlet distribution is utilized to simulate non-IID data across participating entities, with the Dirichlet distribution parameter β is used in this paper, where β is a floating point number. The Dirichlet distribution generates a vector of length N representing the probability distribution over N categories. This vector is denoted as p_k , where $p_{k,j}$

denotes the probability that category k is in party j . Based on this allocation strategy, it is possible that every party may have a limited number (or none at all) of data samples in certain categories.

3.3 Comparison of Precision

In the MOON model, the hyperparameter μ , which governs the model's contrast loss, is calibrated across a range of $\{0.01, 0.1, 1, 5, 10\}$, with the ideal μ values for CIFAR-10, CIFAR-100, and Tiny-Imagenet determined to be 5, 1, and 1, respectively. In the case of FedProx, μ , which influences the proximal term, is adjusted within $\{0.001, 0.01, 0.1, 1\}$, finding the most effective μ values for CIFAR-10, CIFAR-100, and Tiny-Imagenet to be 0.01, 0.001, and 0.001, respectively. These configurations of μ are adopted for the experiments detailed in this study.

After using the above parameter settings, this paper has obtained the test results in Table 1, which prove that for processing non-IID data, the improved methods of MOON and FedProx are effective. Among them, MOON has the highest accuracy, 2.2% higher than FedAvg, and the best improvement. Conversely, FedProx demonstrates only a slight improvement in precision, the proximal term in FedProx does not improve the training performance much, because the value of μ is too small. However, if the value of μ is increased, FedProx will converge too slowly and will not have high accuracy.

Table 1: Average (%) and standard deviation of accuracy for MOON, FedProx and FedAvg.

Method	CIFAR-10	CIFAR-100	Tiny-Imagenet
MOON	67.8±0.3	65.9±0.3	24.5±0.1
FedAvg	65.6±0.4	63.9±0.4	22.3±0.1
FedProx	66.1±0.2	64.1±0.2	22.4±0.1

3.4 Communication Efficiency

Figure 6 shows that MOON, FedProx and FedAvg have almost the same speed of accuracy improvement at the beginning of training, and the model contrastive loss term and proximal term have little impact. Later MOON benefits from the model comparison loss term ℓ_{con} and achieves higher accuracy. At the same time, FedProx does not benefit from the proximal term obviously, because its optimal μ value is too small, making the FedProx precision quite close to FedAvg. However, when $\mu = 1$, precision for FedProx improves too slowly, and it is obvious that FedProx has limited improvement compared with MOON.

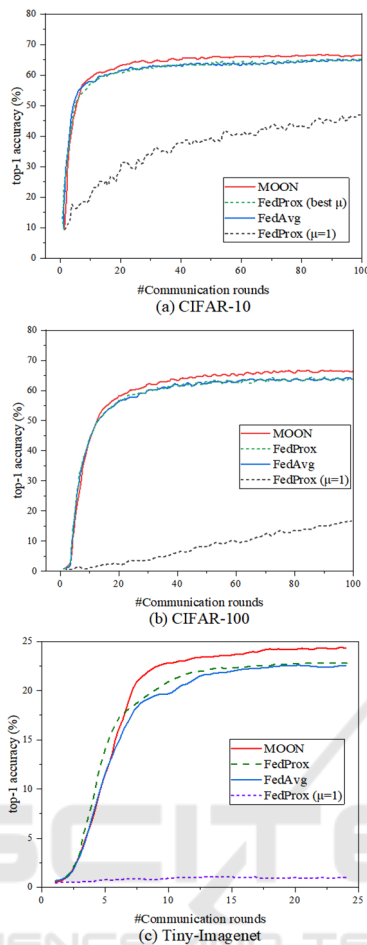


Figure 6: Communication efficiency of different methods.

Table 2 shows the number of rounds required for MOON and FedProx to revery the same accuracy after running FedAvg for 100 rounds on CIFAR-10, CIFAR-100 and Tiny-Imagenet. It can be seen that the communication efficiency of MOON can revery more than twice that of FedAvg, while the communication efficiency of FedProx only increases slightly. MOON's way of communicating is more efficient.

Table 2: Comparison of training rounds different methods to achieve similar accuracy.

Method	CIFAR-10		CIFAR-100		Tiny-Imagenet	
	rounds	speed up	rounds	speed up	rounds	Speed up
FedAvg	100	1×	100	1×	20	1×
FedProx	56	1.8×	77	1.3×	17	1.2×
MOON	28	3.6×	46	2.2×	10	1.8×

3.5 Number of Local Epochs

Figure 7 shows how local epochs' quantity impacts

final model's accuracy across a range of datasets. When there are insufficient local epochs, all methods' local updates are insufficient, training proceeds too slowly, and accuracy is comparatively low. when the count of local epochs exceeds an optimal threshold, local training will drift, that is, the local optimal point and the global optimal point are inconsistent, which makes the precision of all methods decrease. But MOON outperforms the alternatives in all cases.

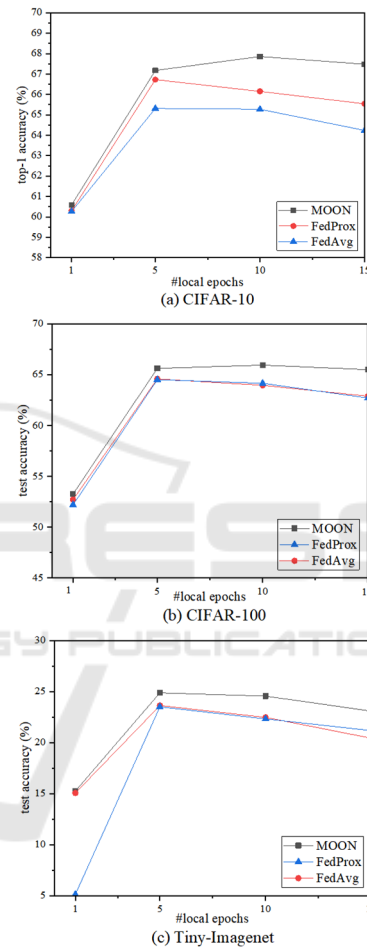


Figure 7: Impact of the amount of local epochs upon the precision of the final model across various datasets.

3.6 Heterogeneity

This paper also investigates the effect of information heterogeneity upon CIFAR-100 through adjustments to the Dirichlet distribution's parameter. A smaller value for leads to a more imbalanced partition. The findings are presented in Table 3, indicating that MOON consistently attains the highest level of accuracy across various non-IID data distributions, while FedProx occasionally performs less effectively

than FedAvg. Experiments show that MOON has better robustness.

Table 3. Precision of MOON, FedProx as well as FedAvg under different non-IID information distributions.

Method	$\beta = 0.1$	$\beta = 0.5$	$\beta = 5$
MOON	62.8%	65.9%	66.3%
FedAvg	61.9%	63.9%	64.9%
FedProx	62.2%	64.1%	64.3%

4 CONCLUSION

Federated learning offers extensive applications and substantial developmental potential, presenting a solution to the issue of data silos originating from a variety of factors. Among them, the processing of non-IID is very important, which greatly affects the final performance of the model. To solve this problem, two methods, FedProx and MOON, have been proposed from the direction of improving local training. Therefore, this paper uses FedProx and MOON to compare the accuracy, communication efficiency, the number of different local epochs and different heterogeneous environments, and shows the superiority of MOON in image classification. It is hoped that these works can help people choose a more suitable model.

REFERENCES

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
<https://www.kaggle.com/c/tiny-imagenet>
- Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2020). A survey on contrastive self-supervised learning. *Technologies*, 9(1), 2.
- Li, L., Fan, Y., Tse, M., & Lin, K. Y. (2020). A review of applications in federated learning. *Computers & Industrial Engineering*, 149, 106854.
- Li, Q., He, B., & Song, D. (2021). Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10713-10722).
- Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2, 429-450.
- Li, X., Huang, K., Yang, W., Wang, S., & Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273-1282). PMLR.
- Yang, Z., Chen, M., Wong, K. K., Poor, H. V., & Cui, S. (2022). Federated learning for 6G: Applications, challenges, and opportunities. *Engineering*, 8, 33-41.
- Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., & Avestimehr, A. S. (2022). Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1), 24-29.
- Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.