An Improved MOON Federated Learning Algorithm Based on Negative Sample Quality Awareness

Sen Yang

{yangsen@glut.edu.cn}

College of Computer Science and Engineering, Guilin University of Technology, Guilin, Nanning, China

Abstract. Federated learning achieves privacy protection through multi-device collaborative training, but model drift caused by the heterogeneity of Non-IID data severely impacts performance. The MOON algorithm uses the previous-round model as negative samples and the global model as positive samples to introduce a contrastive loss to mitigate drift, but indiscriminately rejecting the previous-round model results in the loss of valuable knowledge. This paper proposes an adaptive contrastive federated learning method based on negative sample quality awareness: the rejection weight is dynamically adjusted by calculating the cosine similarity between the previous-round model and the global model—reducing rejection when quality is high to preserve valuable knowledge, and maintaining normal rejection when quality is low to correct bias. In the CIFAR-10 Non-IID scenario, the test accuracy reaches 74.78% (a 0.58% improvement over MOON), the average loss is reduced by approximately 10.09%, and convergence is better, validating the effectiveness of this adaptive mechanism.

Keywords: Federated learning, Model comparison learning, Non-IID data, Adaptive weights, MOON.

1 Introduction

In recent years, IoT devices have become increasingly widespread and integrated into all aspects of life. Edge computing, which has developed alongside this trend, has gradually become a way to process massive amounts of data. It moves computing from the cloud to near the data generation point, improving performance in terms of bandwidth, latency, privacy, and reliability. In the IoT field, massive amounts of high-frequency data are generated by video and industrial sensors. Uploading all this data back to the cloud would not only cause bandwidth congestion and high transmission costs, but not all scenarios require data backhauling. Many tasks are better suited for local real-time processing followed by uploading and summarizing the results. In other scenarios such as autonomous driving and industrial control, where millisecond-level response times are critical, cloud round-trip latency and jitter are extremely sensitive; edge computing can significantly improve real-time performance [1].

ICIAAI 2026, May 22-24, Singapore, Singapore

DOI 10.4108/eai.22-5-2026.2365235

Due to compliance and confidentiality requirements, some restricted sensitive data should not be uploaded centrally; edge processing helps reduce the risk of data leaving the network. In weak network or disconnected environments, edge deployment of critical inference and early warning systems can also improve system availability and robustness[2, 3]. Edge computing utilizes nearby devices for computation, reducing communication pressure and improving real-time performance, privacy, and reliability [4].

Among them, federated learning has gradually become the main method[5-7]. With its development, the emergence of the Moon algorithm has brought a way to speed up the computation [8]. MOON effectively alleviates the client drift and convergence instability caused by Non-IID in federated learning: by introducing model-level contrastive learning in local training, the global/historical model is used as the contrastive constraint representation for updating, reducing the inter-round offset, making the local model closer to the global semantic space, thereby improving training stability and generalization performance[8, 9]. However, in the later stages of computation training, if the results obtained in the previous round of training are generally not too far off, but in the learning process, no matter what the results are, they are all treated as negative samples, which may produce a negative problem. For example, the results obtained from training are not bad, but they are discarded, and the optimal solution cannot be obtained or catastrophic forgetting is aggravated.

To address this problem, this paper optimizes and improves the Moon algorithm by introducing a discriminative component. During computation, before discarding or classifying a sample as negative, the performance of the current model is compared. If the performance is good, the sample is not considered negative, and training continues based on it. Conversely, if the performance is poor, the sample is discarded, and a new model is trained. Experiments show that this improved method significantly outperforms the original Moon algorithm for some datasets.

2 MOON Optimization and Implementation

2.1 Overall process of federated learning

Federated learning utilizes edge devices in the network, excluding core cloud service providers and network transmission equipment. The server may or may not distribute the initial global model, and local clients use local user data to train the model. The training results are then sent back to the server to further improve the server's model. This approach not only protects user privacy but also keeps the user's dataset on their device, preventing it from being uploaded to the server, thus improving the model in a win-win manner. [10]

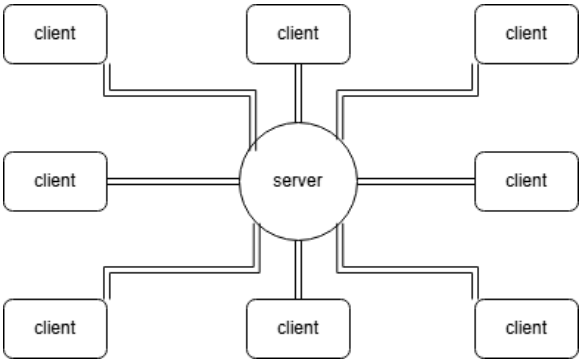

Fig. 1. Federated learning involves equipment diagrams (Picture credit: Original)

As shown in Figure 1, Federated learning typically uses a server and multiple clients, with a workflow divided into four parts. First, the server generates an initial model and distributes it to the clients participating in the training. Second, the local training process begins on each client after receiving the initial model and training based on their local data. After training, the updated parameters of the model are obtained. Third, the upload and update process involves communication between the client and server, uploading only the parameters of the trained model, not the original data. Finally, the server, upon receiving the models from each client, performs a global aggregation, usually calculated using a weighted average, to obtain a new global model.

The formula is as follows: $\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \theta_{t,k}$, where $n_k$is the amount of data from the k-th client, and θis the model parameters. The above steps are executed iteratively. The server then returns to step 1 and redistributes the aggregated model to each client, and each client continues to repeat the above steps.

The most basic FedAvg algorithm involves the server distributing an initial model [4], each edge device calculating the model, and the calculated model being sent back to the server. The server then calculates the average of the collected models. The specific steps include initialization, local training, uploading and updating, global aggregation, and loop execution. The aggregation formula of FedAvg is: $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^{k}$, where n_k refers to the number of samples owned by the kth client, n refers to the total number of clients participating in this round of training, andω_{t+1}^{k} refers to the weights of the local model obtained after the client training. However, in reality, due to the different preferences of each edge device and the different preferences of each user, the results obtained by each client vary greatly, and the results obtained after averaging back to the server are not ideal. The Moon algorithm, based on FedAvg, mainly optimizes and improves by moving away from the model trained locally in the previous round to fix the problem of unsatisfactory performance.

It mainly adopts comparative learning. The samples used in the Moon algorithm are the sample models distributed by the server. These sample models are compiled from data from multiple users and set as samples to ensure that the model calculated by each device is relatively similar to the server's model. Simultaneously, it aims to avoid closely resembling

models trained in previous rounds, preventing the algorithm from deviating from its intended path.

In each round of computation, Moon's edge devices run three models: a public model distributed by the server, a model obtained from the previous round, and a model used in the current round. Moon's characteristics require the current round's model to closely resemble the public model and distance itself from the previous round's model. The server-distributed model is not modified. Therefore, for the same sample, it outputs three feature vectors: z(current round), z(official), and z(previous round). This learning method allows for large-scale consistency with the server, while allowing inconsistencies in other data.

This paper argues that there is a problem here. Typically, in the later stages of training, the results from the previous training round are not significantly off-target. The original logic treats all results, regardless of their quality, as negative samples. This leads to a negative problem: if the training results are not actually bad but are discarded, the optimal solution cannot be obtained, or catastrophic forgetting may occur.

2.2 Improved MOON overall structure and process

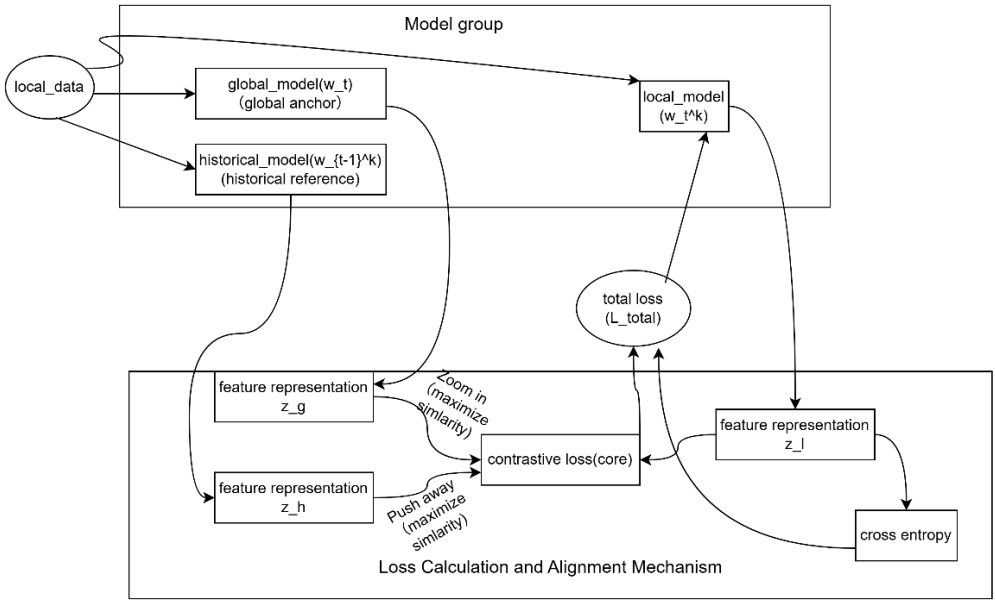

Fig. 2. Original Moon model flowchart (Picture credit: Original)

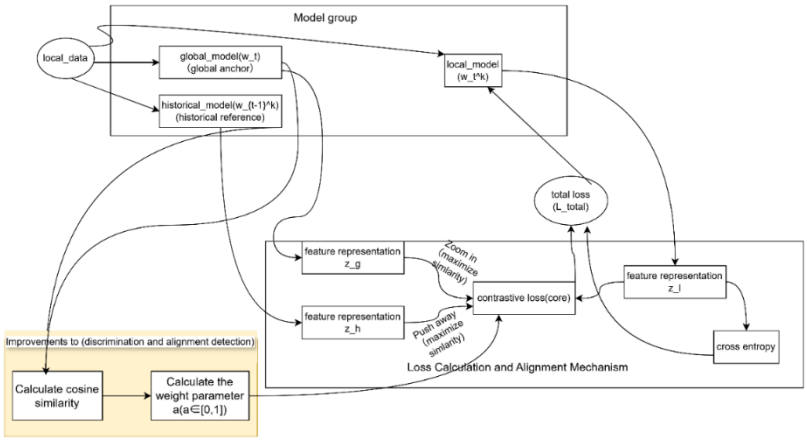

Fig. 3. Improved flowchart (Picture credit: Original)

Figure 2 and Figure 3 show the original Moon model flowchart and improved flowchart. The consistency level is represented by the yellow module in the diagram. It determines whether the current model deviates from the global model by calculating the cosine similarity between the global model and historical models. During the detection process, it is compared with the main model distributed by the server. If the similarity is high, it indicates that the training direction is not off track, and a rejection weight is introduced. If there is no deviation, the rejection weight is reduced. If the difference is high, it indicates a significant deviation, and the rejection weight is increased for normal rejection. Based on the above characteristics, an adaptive weight parameter a can be given as the decision center: when the similarity performs well, then a is closer to 0, indicating that negative samples are discarded and training continues on the basis of the historical model. In this case, they are regarded as negative samples, and the next round of training will develop towards deviating from the local model.

2.3 Loss function optimization

The total loss is calculated using the formula

$$L_{total} = L_{\text{sup}}(\text{Supervised Loss}) + \mu \cdot L_{\text{con}}(\text{Contrastive Loss}) \quad (1)$$

The goal is to minimize$\mathrm{L}_{\text{total}}$ as much as possible (similar to general deep learning, a smaller L_{sup} results in a better model). The supervised lossL_{sup} is mainly the loss used during the training of classification models, such as cross-entropy loss. Its function is to ensure the model correctly classifies the input data. It is calculated by each client using its local data to perform calculations, predicting the label, and then comparing the predicted label with the true label. The contrastive loss L_{con} is the loss generated by contrastive learning. It is used by Moon to address the issue of different data distributions among clients (Non-IID). Its function is to make the local model more closely resemble the global model, preventing the local model from simply learning from itself. The model and the global model extract features from the same batch of data, then calculate the feature similarity between the current model and the global model, and compare it with the feature similarity of the local model obtained in the previous learning round. A contrastive loss formula is used to try to make the local model

closer to the global model while maintaining a distance from the model learned locally in the previous round. The parameter μ controls the weight of the contrastive loss, dynamically changing according to whether it leans more towards the global model. When the deviation between the current round's result and the global model is relatively large, causing the local model to develop more in the direction of the global model, μ is reduced. The local model is basically unaffected and continues to develop based on the local data.

The contrastive loss formula is as follows:

$$L_{con} = -\log \frac{\exp\left(\frac{\text{sim}(z, z^{glob})}{\tau}\right)}{\exp\left(\frac{\text{sim}(z, z^{glob})}{\tau}\right) + \exp\left(\frac{\text{sim}(z, z^{prev})}{\tau}\right)} Z \tag{2}$$

represents the features extracted by the current model. Z^{glob} represents the features extracted by the global model.Z^{prev}represents the features extracted by the local model in the previous round. It can be understood as

$$\text{P}_{\text{probability}} = \frac{\exp(\text{Positive samples})}{\exp(\text{negative samples}) + \exp(\text{Positive samples})} \tag{3}$$

then take -log, which becomes a number that is closer to 1 the better.

3 Experiment

3.1 Experimental Environment

Table 1. Hardware Parameters

CPU	AMD Ryzen 7 4800H (8 cores, 16 threads, 2.9GHz)
GPU	NVIDIA GeForce GTX 1650 Ti (4GB GDDR6 memory)
RAM	16GB (DDR4 3200MHz)
Disk	512GB NVMe SSD

Table 2. Software Parameters

Device	cpu
Number of Clients	10
Number of Communication Rounds	20
Number of Local Training Rounds	5
Participation Ratio per Round	0.3
MOON Contrast Loss Weight (μ)	1.0
Temperature Parameter (τ)	0.5
Data Heterogeneity (α)	0.5

The hardware and software parameters of the experimental environment are shown in Tables 1 and 2. First, the dataset was divided into Non-IID (non-uniform) partitions using a Dirichlet distribution. Then, a proportion parameter was generated for each device based on the uniformity parameter. Multiplying the total number by the proportion yielded the expected number of numbers for that device, avoiding an idealized scenario where the numbers for each

device are uniformly distributed, thus providing data closer to real-world conditions. The alpha parameter controls the uniformity of the proportion parameter for each device; a larger alpha parameter indicates a more uniform proportion for each device. After multiple runs, the results are summarized in the following line graph（Figure 4）.

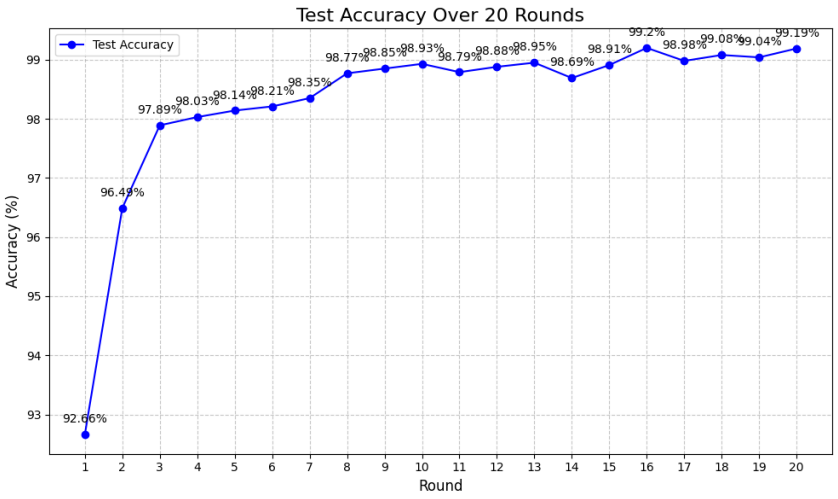

Fig. 4. Test Accuracy Line graph (Picture credit: Original)

The overall trend shows a rapid increase to 97.89% in the first three rounds, indicating the effectiveness of the initial aggregation. Between rounds 4 and 10, the accuracy gradually increased from 98.03% to 98.93%, with the growth rate slowing down. After rounds 11-20, the accuracy fluctuated slightly but remained at a high level overall.

The original formula was

$$L_{con} = \frac{-\log(\exp\left(\frac{\mathrm{sim}(z, z_{\mathrm{glob}})}{\tau}\right))}{(\exp(\mathrm{sim}(z, z_{\mathrm{glob}})/\tau) + \exp(\mathrm{sim}(z, z_{\mathrm{prev}})/\tau)))} \tag{4}$$

by improved formula introduces an adaptive weight 'a':

$$L_{con} = \frac{-\log(\exp\left(\frac{\mathrm{sim}(z, z_{\mathrm{glob}})}{\tau}\right))}{(\exp(\mathrm{sim}(z, z_{\mathrm{glob}})/\tau) + a \times \exp(\mathrm{sim}(z, z_{\mathrm{prev}})/\tau)))} \tag{5}$$

In the initial experiment, the default parameter adjustment was too aggressive, automatically increasing the parameter if the match was small, and the parameter range was set unreasonably. In the improved version, a minimum weight was added, defaulting to 0.3, to ensure a certain degree of repulsion.

This is then mapped to the range [a_min, 1.0]. When there is complete repulsion, $\mathrm{sigmoid} = 1$, i.e., $a = 1.0$. When minimizing repulsion, $\mathrm{sigmoid} = 0$, i.e., $a = a_{min}$. Here,

$sigmoid = \frac{1.0}{(1.0 + \text{np.exp}(\text{sensitivity} \times (\text{consistency} - \text{threshold})))}$, and a is obtained by calculating $a_{\min} + (1.0 - a_{min}) \times sigmoid$

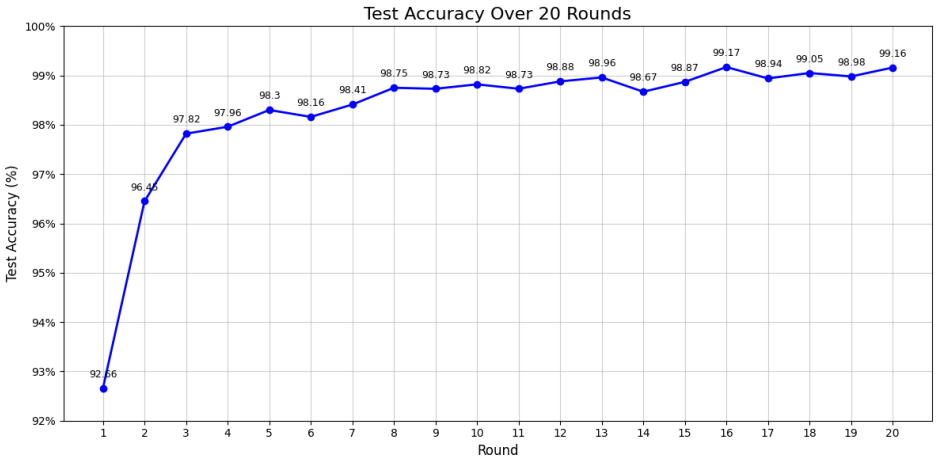

Fig. 5. Line chart accuracy rates across 20 rounds of testing (Picture credit: Original)

The improved solution achieved a final test accuracy of 99.16% (Figure 5). Compared to the 99.18% accuracy of Moon without any parameters and FedEx, this result is within the error range, raising doubts about its validity. Running the original Moon solution again yielded unsatisfactory results. Considering the potential error, the comparison model was found to be faulty. After modification, the dataset was switched to CIFAR10, and the solution was run again. The results are as Figure 6.

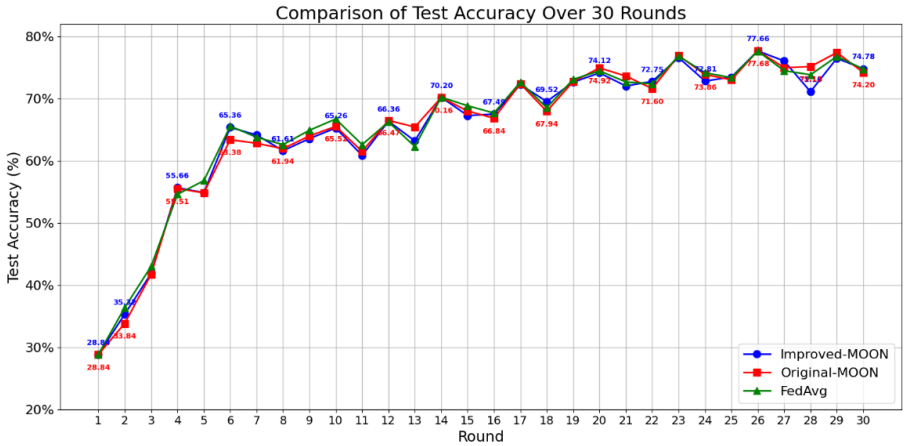

Fig. 6. Line chart comparing accuracy rates across 30 rounds of testing (Picture credit: Original)

Table 3. This summary outlines the key metrics for the three models on the CIFAR10 dataset

Metrics	FedAvg	Original-MOON	Improved-MOON
Final Accuracy（%）	74.75%	74.20%	74.78%
Highest Accuracy（%）	77.62%	77.68%	77.66%
Average Accuracy in the Last 10 Rounds（%）	74.68%	74.84%	74.37%
Standard Deviation of Accuracy in the Last 10 Rounds（%）	1.73%	1.89	2.13
Rounds Reaching 65% for the First Time	R6	R10	R6
Average Training Loss in the Last 10 Rounds	0.2685*	0.44 • 52	0.4003

Table 4. The statistics of the improved adaptive weighting mechanism of MOON are shown.

Indicator	Values
Model Consistency Mean	0.7031
Model Consistency Range	[0.3019, 0.7935]
Adaptive Weight α Mean	0.7949
Adaptive Weight α Range	[0.5548, 0.9429]
Number of times α < 0.7	6/29
Number of times α > 0.9	3/29

Comparing the final accuracy rates. Table 3 shows that the three methods achieve similar final accuracy rates after 30 runs. The improved MOON achieves 74.78% accuracy, the original MOON achieves 74.20% accuracy, and FedAvg achieves 74.75% accuracy. Compared to the original MOON, the improved MOON represents an improvement of +0.58%, and compared to FedAvg, the improved version represents an improvement of +0.03%. Round 26 is the round in which the highest accuracy of the three methods appears. The small difference in the values indicates that in this improved scenario, the upper bound of all models changes less. The main differences are reflected in the stability and convergence characteristics of the training process.

Comparing convergence speeds, the three methods converged at roughly the same rate in the early stages of training. By the fourth round, all three models had surpassed 50%, and by the 14th round, they had reached 70%. Notably, at the rounds where they first reached 65%, FedAvg and the improved MOON achieved 65.36% and 65.56% respectively by the 6th round, while the original MOON only reached 65% by the 10th round. This indicates that some factors in the original MOON model, such as excessive constraints, slowed down the operation, while the improved MOON effectively optimized this problem through an adaptive mechanism.

Comparing training loss, because FedAvg and MOON calculate the loss differently (MOON's calculation is$L_{total} + L_{CE} + \mu \times L_{con}$), the comparison is only between the two MOON models, primarily focusing on the negative sample weights 'a'. Experimental results show that

the improved MOON has an average training loss of 0.4003 in the last 10 training rounds, which is about 10.09% lower than the original MOON (0.4452), demonstrating the effectiveness of the adaptive weight mechanism in reducing training loss.

Table 4 shows that the adaptive weights exhibit significant dynamic adjustment during training. The average weight 'a' is 0.7949, lower than the fixed average of 1.0 in the original MOON, indicating that the adaptive weights reduce the rejection strength of negative samples in most cases. Statistics show that in 6 out of 29 rounds, the value of 'a' was below 0.7, and in 3 rounds, the value of 'a' exceeded 0.9, close to the original MOON parameters, indicating that the adaptive weights can dynamically adjust according to the degree of difference.

Discussion Summarizing the above experimental results, it can be found that the fluctuations in all performance differences in the current training results are not significant. Possible reasons include: The number of training rounds is relatively small. This experiment ran for 30 rounds. The results may not fully reflect the differences among the three methods. Increasing the number of rounds might reveal more differences, such as the excessive rejection by MOON and the lack of constraints by FedAvg, which would further amplify the impact. The data heterogeneity was moderate. The data heterogeneity parameter α in this experiment was set to 0.3, corresponding to a moderate level of Non-IID scenario, which may not have revealed more significant specific differences. In some extreme scenarios, such as when α is 0.1, the differences would be further amplified.

Although the fluctuation in the final accuracy was not significant, it improved the reduction of training loss and the early convergence speed, indicating the effectiveness of adaptive weights.

3.2 Discussion

Observing the line graph, some characteristics can be found: the adaptive weights have little effect on improving the results, but have a significant impact mainly on the training phase. The difference in the final accuracy of the three methods did not fluctuate much, indicating that FedAvg can achieve a better performance level. From the training process, MOON exhibited a different trend compared to FedAvg. The original MOON model only reached 63.38% in the sixth round, while FedAvg and the improved MOON reached 65.56% and 65.36% respectively during the same period, higher than the original MOON. This indicates that fixed strength introduces additional overhead.

The optimization of adaptive weights is mainly reflected in the efficiency during training, rather than in the final result. Although the improved MOON only achieved a 0.58% improvement over the original MOON, it reduced the training loss by 10.09% in the last 10 training rounds. This indicates that the adaptive weight mechanism optimized the model by controlling the repulsion coefficient α, reducing unnecessary conflicts and achieving a near-perfect accuracy with a lower loss rate.

4 Conclusions

This paper, based on federated learning, improves the original MOON model by adding adaptive weight parameters to dynamically adjust the repulsion coefficient α, thereby reducing unnecessary conflicts and effectively lowering the training loss. Specifically, using the

CIFAR10 Non-IID dataset with α=0.3, the improved MOON slightly outperformed the original MOON (74.20%) and FedAvg (74.75%) with a final accuracy of 74.78%, proving the effectiveness of the adaptive weights.

The main advantage of the improved MOON lies in improving the efficiency of the training phase. By dynamically adjusting the repulsion coefficient based on the results of the previous round of local model training, the training loss was effectively reduced by 10%, while avoiding the training sluggishness problem caused by the fixed repulsion coefficient in the middle training stage of the original MOON.

References

[1] Li, Qinbin, Bingsheng He, and Dawn Song. "Model-contrastive federated learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[2] Liu, Fang, et al. "A survey on edge computing systems and tools." Proceedings of the IEEE 107.8 (2019): 1537-1562.

[3] Cao, Keyan, et al. "An overview on edge computing research." IEEE access 8 (2020): 85714-85728.

[4] Hou Jin. Design of resource scheduling algorithm for edge nodes [J]. Fujian Computer, 2026, 42(02):22-26. DOI:10.16707/j.cnki.fjpc.2026.02.004.

[5] Yang Zhihan. Federated learning optimization algorithm based on contrastive learning [J]. Information and Computer, 2025, 37(11):1-3.

[6] Li Xiangxiang. Research and application of image classification based on federated learning [D]. Jiangnan University, 2025. DOI:10.27169/d.cnki.gwqgu.2025.001835.

[7] Chen Hao. Research on personalized heterogeneous federated learning algorithm based on deep mutual learning [D]. Changchun University of Technology, 2025. DOI:10.27805/d.cnki.gccgy.2025.001267.

[8] Li, Qinbin, Bingsheng He, and Dawn Song. "Model-contrastive federated learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[9] Chen, Changyu, and Weiheng Rao. "Optimization of Moon Model-Contrastive Federated Learning." In Proceedings of the 2nd International Conference on Data Science and Engineering (ICDSE 2025), pages 128-132

[10] Li, Yanrui. "An efficient communication method based on model contrastive learning and model pruning." Third International Conference on Big Data, Computational Intelligence, and Applications (BDCIA 2025). Vol. 14128. SPIE, 2026