PFL-NON-IID Framework: Evaluating MOON Algorithm on Handling Non-IID Data Distributions

Sheng Chen[1], Jiancheng Peng[2], Andi Tong[3,*] and Cong Wu[4]

[1]School of Computer Science and Technology, Donghua University, Shanghai, 201612, China
[2]Sino-European School of Technology, Shanghai University, Shanghai, 200444, China
[3]College of Electronics and Information Engineering, Shenzhen University, Shenzhen, 518040, China
[4]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China
*2020282067@email.szu.edu.cn

Abstract. This paper introduces an optimized version of the MOON algorithm for federated learning, which is a distributed machine learning approach designed to tackle challenges related to data privacy and decentralization. However, the performance of traditional federated learning methods is hindered by the non-uniform distribution of data across nodes, a problem known as the Non-Identical and Independently Distributed (NON-IID) problem. The MOON algorithm leverages statistically heterogeneous data for personalized model training and improves the issues of slow gradient descent rates and high network communication overhead. Although the MOON algorithm has shown promising results, it still faces challenges in terms of computational complexity and training efficiency. Therefore, this paper aims to further optimize the MOON algorithm to enhance its computational efficiency and training effectiveness. The paper implements the MOON algorithm using the PFL-NON-IID framework and verifies its effectiveness in handling non-uniform data distributions. It also analyzes and compares the experimental results to optimize the algorithm's computational performance, real-time capability, and scalability. The study aims to provide support for the application of distributed deep learning systems.

Keywords: Deep learning, Moon, Data models, Federated learning, Training effectiveness

1 Introduction

Federated Learning is an approach to distributed machine learning that aims to tackle challenges related to data privacy and decentralization. It enables model training without sharing raw data, thus protecting user privacy [1]. Federated Learning has gained considerable attention for its ability to perform centralized learning without compromising data privacy. It is a decentralized learning technique that reduces the risk of data leakage by decomposing the dataset and algorithm's execution status, allowing distributed user devices to collaborate on model training without disclosing sensitive data to other participants. Federated Learning enables large-scale training on devices that generate data while keeping sensitive data retained by data owners through local data collection and training. The core entities involved in Federated Learning are the central server and client devices, where the server coordinates the clients at a macro level, and the clients perform local machine learning model training.

The general workflow of Federated Learning involves several steps. First, each client independently trains on its local data. Next, the clients encrypt the model parameters or gradients and upload them to the server. The server securely aggregates the collected client model parameters or gradients to ensure secure aggregation. Finally, the server distributes the model to the corresponding clients. This workflow enables collaborative model training while preserving data privacy in a distributed environment.

However, in practical applications, we face a significant problem: the data distribution across different nodes is highly uneven, known as the Non-Identical and Independently Distributed (NON-IID) problem [2]. This non-uniform data distribution hinders the performance of traditional federated learning methods when dealing with non-uniform data. These methods fail to adequately consider the data feature differences among nodes, thereby impacting the model's generalization ability. Additionally, the non-uniform data distribution can lead to poor training performance on certain nodes and introduce bias during the model parameter aggregation process, thereby degrading the overall model performance.

To address the NON-IID problem, researchers have proposed the MOON algorithm [2]. The MOON algorithm is a model comparison-based federated learning algorithm designed to leverage statistically heterogeneous data for personalized model training. This algorithm effectively tackles the challenge of non-uniform data distribution by progressively fusing local models from multiple nodes and improves the issues of slow gradient descent rates and high network communication overhead. Experimental results demonstrate that the MOON algorithm exhibits remarkable privacy protection while outperforming traditional methods in terms of speed and accuracy [2].

Nevertheless, the MOON algorithm still faces challenges in practical applications. The extensive model training, parameter transmission, and computations involved in the MOON algorithm result in high computational complexity, long training times, and hampered real-time performance and scalability. Therefore, it is crucial to further optimize the MOON algorithm to enhance its computational efficiency and training effectiveness.

Therefore, this paper aims to experimentally and practically implement the MOON algorithm using the PFL-NON-IID framework. The objective is to verify the effectiveness and feasibility of the MOON algorithm in handling non-uniform data distributions. Additionally, through the analysis and comparison of experimental results, this research aims to explore and optimize the computational performance and training efficiency of the MOON algorithm, thereby enhancing its real-time capability and scalability [3]. Ultimately, this study aims to provide further support and assurance for the application of distributed deep learning systems.

2 Related Work

In this section, we present the background of Federated Learning and related research work.

2.1 Background

Over the last few years, a lot of research has been conducted focusing on the impact of data heterogeneity on distributed learning. By way of example, in [4] the author proposed the Federated Multi-Task Learning (FedMT) method, which can handle the problem of different data distributions on each node, but has limited effectiveness in dealing with data heterogeneity. Zhao et al. [5] proposed a distributed heterogeneity adaptation method by analyzing the differences in data distributions of different nodes, which is improving the accuracy of the model, but still suffers from slow convergence. In [6], the author proposed an adaptive Federated learning method in a resource-constrained edge computing system, which improves system efficiency by dynamically adjusting device participation and local iterations but may not perform well in an unstable network environment. In [7], the author proposed a Federated learning method for heterogeneous data by introducing a hybrid update strategy, which can personalize the model under different data distributions but may require more communication rounds. A personalized Federated learning method based on meta-learning is proposed in [8]. A meta-learner is trained to adapt to the data distribution of different users, which improves the personalized performance of the model, but its computational complexity may be high. In addition, the author of [9] proposed an efficient Federated learning method designed for heterogeneous clients, which shows advantages in computing and communication efficiency but may pose challenges when dealing with extremely heterogeneous environments. Comprehensive research[10] was conducted on Federated learning in the Internet of Things, providing rich research background and the latest progress for the outside world, but no specific solutions were proposed for specific problems.

2.2 MOON algorithm

In this context, the MOON algorithm emerged. It proposes a method based on model comparison, which uses the personalized information in the statistical heterogeneous data to train the personalized model, to better deal with the data heterogeneity in Federated learning. Through model training on local devices, the updated model parameters are aggregated to the central server in Federated learning to achieve the global update of the model. However, the traditional Federated learning algorithm has some challenges when dealing with statistical heterogeneous data. The MOON algorithm proposes a model comparison-based method that utilizes personalized information from statistical heterogeneous data to train personalized models.

2.3 PFL-NON-IID platform

We use the PFL-NON-IID platform to implement the MOON algorithm. PFL-NON-IID is a deep learning framework based on PyTorch, providing a new distributed data processing method that eliminates the problem of data sample distribution between different nodes. It provides the PFL (Privacy Resilient Federated Learning) algorithm, which enables multiple devices to communicate and train models with each other without disclosing data, thereby achieving the goal of improving model performance. The PFL-NON-IID platform has the advantages of strong distributed computing power, strict data privacy protection, and efficient and simple implementation, providing strong support for the application of distributed deep learning.

3 Methodology

3.1 Introduction of MOON algorithm

The MOON (Mixture Of Online Learning) algorithm [11] employs a distributed approach to machine learning, specifically designed for training a global model within the framework of Federated Learning. By comparing the features generated on the same data across different models, the local training update directions are corrected.
Introduction to the Process of the MOON Algorithm：
Initialization of global model parameters, w, and participant model parameters, w_i, is performed initially. For each training round, denoted as k = 1, 2, ..., K:
1. The central server disseminates the model parameters, denoted as "w," to all participants involved in the process.
2. Each participant performs the following operations locally:
Compute the Gradient Direction Similarity (GDSi): Participants use their local dataset to calculate the gradient of their current model parameters, w_i, and measure the difference between this gradient and the gradient of the global model parameters, w. This difference can be quantified using metrics such as cosine similarity or other similarity measures.
3. Calculate the weight, λ_i, based on the Gradient Direction Similarity (GDSi): Participants determine their weight using a formula or rule that takes into account the gradient direction similarity. The weight represents the contribution of the participant to the global model, and participants with higher similarity may have larger weights.
4. Perform T rounds of gradient descent on their local dataset to update their model parameters, w_i: Participants execute T rounds of the gradient descent algorithm using their local dataset. They calculate gradients based on the local data and update their model parameters, w_i, accordingly.
5. Send their model parameters, w_i, and weights, λ_i, to the central server: Participants transmit their updated model parameters, w_i, and the computed weights, λ_i, to the central server for further processing.
6. The central server selects some part of the participants based on a certain rule (e.g., weighted average) upon receiving the model parameters, w_i, and weights, λ_i, from the

participants. It then performs a weighted average of the selected participants' model parameters to obtain the updated global model parameters, w_i.

7. After the K training rounds, the MOON algorithm returns the final global model parameters, w_i, as the training result. These parameters represent the optimized model obtained through the MOON algorithm's iterative process using the distributed learning framework.

3.2 Advantages of the MOON algorithm

Prevent Data breach: Moon algorithm adopts Federated learning architecture so that every client's data can be stored locally and will not be transmitted to the centralized server, thus protecting data privacy.

Reduce computational overhead: The Moon algorithm uses a new feature aggregation-based model preprocessing method, which allows the global model to be decomposed into local models during model training, thereby reducing computational complexity and training time.

Improve algorithm accuracy: Moon algorithm can improve algorithm accuracy through fast learning and data accumulation. The Moon algorithm supports online learning, allowing the model to be updated in real-time applications, thereby enabling the algorithm to adapt to changes in different scene environments.

Distributed storage support: The Moon algorithm distributes the training weights of the model across multiple machines, making it scalable and supported by distributed storage. This means that the Moon algorithm can be used to process large-scale data and can quickly scale to support more clients.

3.3 The flow of the MOON algorithm

1. Initialize the global model parameters w and the model parameters w_i for each participant. For each round of training k=1,2,..., K: The central server broadcasts the global model parameter w to every participant;

2. Each participant performs the following actions locally:

(1) Calculate one's gradient direction similarity GDSi;

(2) Calculate its weight based on GDSi λ_i;

(3) Perform a T-round gradient descent on the local dataset and update one's model parameters w_i;

(4) Set one's own model parameters w_i and weights λ_i and send it to the central server;

3. The central server selects a portion of participants based on a certain rule and weights their model parameters to update the global model parameter w.

4. Return the final global model parameter w.

The Logical framework of moon algorithm and its specific part description are shown in Fig. 1.

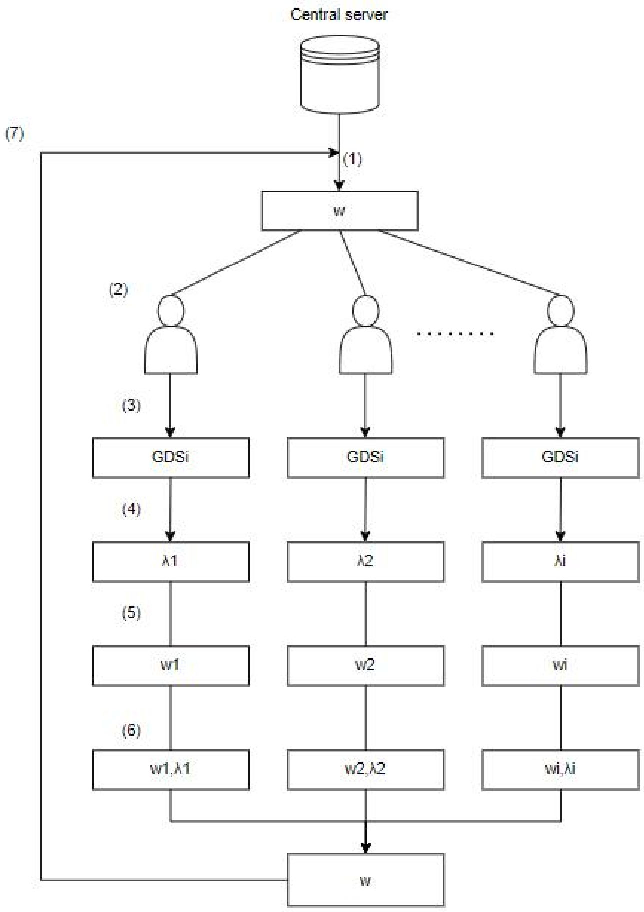

Fig. 1. The block diagram of the moon algorithm. (Picture credit:Original)

（1）Initialization of Global Model Parameters, w, and Participant Model Parameters, w_i

（2）For each training iteration, the central server disseminates the model parameters, denoted as "w," to all participants.

（3）Each participant computes the Gradient Direction Similarity (GDSi) locally.

（4）Each participant calculates their weight, λ_i, based on the Gradient Direction Similarity (GDSi).

（5）Each participant performs T rounds of gradient descent on their local dataset to update their model parameters, w_i.

（6）The updated model parameters, w_i, and weights, λ_i, are sent to the central server by each participant.

（7）The central server applies specific selection rules to choose a subset of participants, and then conducts a weighted averaging of their model parameters to update the global model parameters, denoted as "w_i".

4 Experiments

4.1 Experimental Setup

We will use a pathological non-IID dataset as the experimental dataset and evaluate the effectiveness of the MOON algorithm by comparing its performance with other benchmark algorithms in terms of accuracy and personalization performance. The experiments will consider different cases of statistical heterogeneity and several devices and analyze the performance variation of the MOON algorithm in these cases.

4.2 Comparison of the contents

We experimented with the hyperparameters such as the number of partial iterations, batch size, learning rate, the proportion of training clients involved, and several local iterations using appropriate data sets that can lead to a steady increase in their accuracy rates, respectively, to find the relatively optimal hyperparameters. In our experiments, we use the control variables method to compare the effect of the different values of each variable on the accuracy rate by making a cross-sectional comparison.

To address the statistical heterogeneity problem and thus be able to obtain higher accuracy on local datasets, we reduce this heterogeneity by adjusting the aggregation weights. We include in the source code three additional different ways of assigning aggregation weights:

1. Use the same weight for each client
2. The appropriate aggregation weight is assigned according to the client's training loss, the smaller the loss, the greater the aggregation weight.
3. Assign the corresponding aggregation weight based on the accuracy of the client, the greater the accuracy the greater the aggregation weight.

By learning these three different aggregation weighting approaches with default parameters, we can compare their accuracy gaps to observe the impact of different aggregation weights on heterogeneity.

4.3 Optimization measures

Based on the basic implementation of the MOON algorithm, we will conduct further optimization studies. Due to the specificity of pathological non-IID datasets, we can adopt specific optimization strategies to improve the performance of the algorithm. The specific optimization strategies may include adjusting the model architecture, introducing regularization methods, improving the model comparison strategy, etc. Through these optimization measures, we hope to further improve the effectiveness of the MOON algorithm in processing pathological non-IID datasets. We first optimized the hyperparameter part, and we found that changes in four of the hyperparameters have a significant impact on the accuracy. As shown in Fig. 2, the graph of a change in learning rate does not result in a significant change in accuracy in the range of 0.01 to 0.001. In Fig. 3, we discover that as the number of clients increases, there is a slight increase in accuracy at first, but a significant decrease in accuracy from more than five clients onwards. The chart in Fig. 4 indicates that that the accuracy remains at a high level until the number of local epochs is 10, but after 10 the accuracy drops and remains at a low level. By observing the chart in Fig. 5, the variable μ reaches its maximum accuracy at 0.5, while other values in the range of 0.1 to 10 correspond to a slightly lower accuracy.

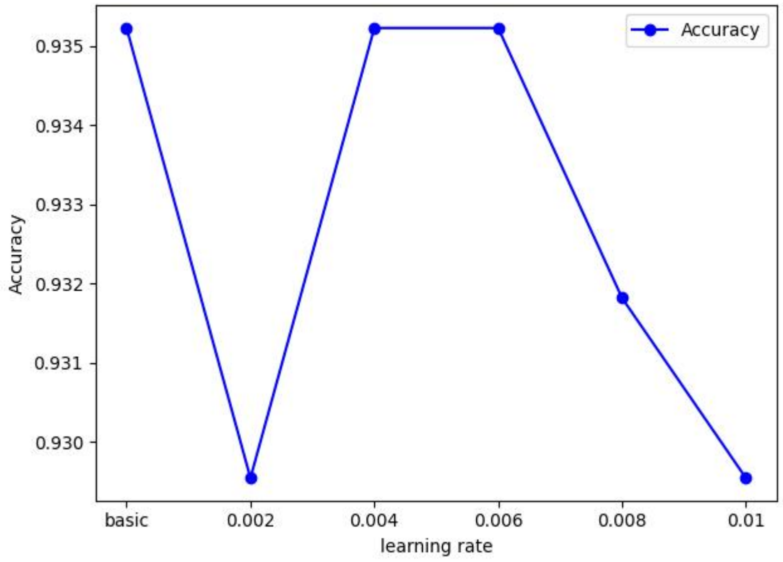

Fig. 2. Model Accuracy with Learning rate. (Picture credit:Original)

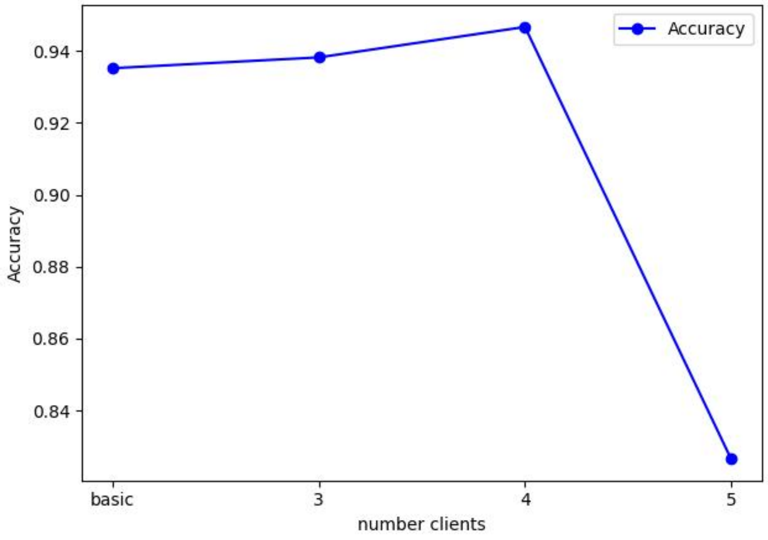

Fig. 3. Model Accuracy with number clients. (Picture credit:Original)

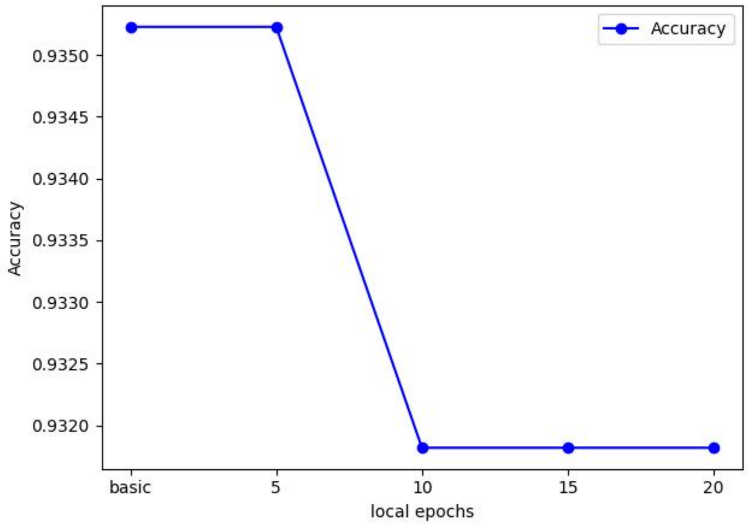

Fig. 4. Model Accuracy with local epochs. (Picture credit:Original)

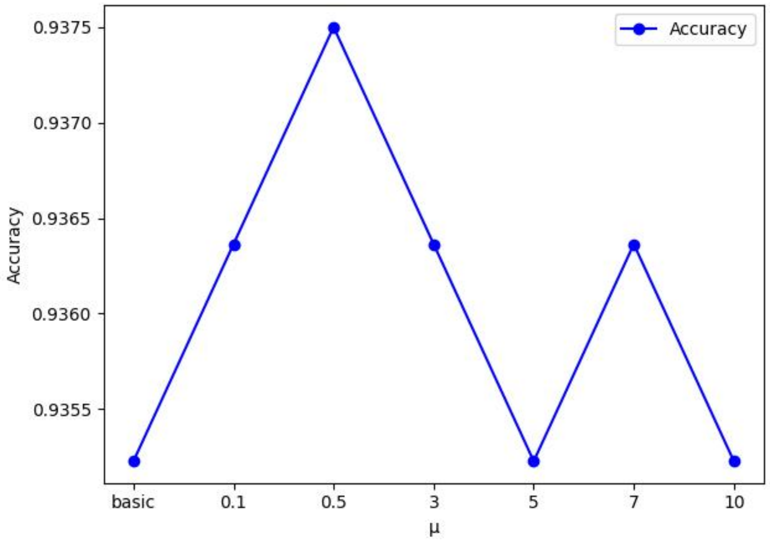

Fig. 5. Model Accuracy with variable μ. (Picture credit:Original)

The default value for the learning rate is 0.005 and by varying the learning rate we can see that the local learning rate is more accurate around the default value. The default value for the total number of clients is 2. By varying the total number of clients we can see that higher accuracy is achieved when the number is around 4, and that accuracy decreases significantly after 5. The default value for the number of iterations is 1. By varying the number of iterations, it can be seen that the accuracy rate is higher below 10 and hurts the accuracy rate after 10. The default value for the control variable μ is 1. By varying μ, we can see that the highest accuracy is achieved when μ is close to 0.5.

Previous studies used $\mu = [0.1, 0.3, 0.5, 0.7, 1]$ to test the performance of the model in CiteSeer. Among them, The best accuracy rates for FedAvg and MOON were 70.6% and 69.8% respectively, while MOON was less robust in node classification, with accuracy rates fluctuating between 57.7% and 69.8%. This indicates that the weighted similarity-constrained approach improves convergence and robustness. This study also found that lower beta parameters enable better performance of the model with more effective performance [12].

For the aggregation weights, in addition to the source code default (assigning weights based on the size of the dataset), three other methods of assigning weights were used. The first approach is to assign weights equally to each client. The second way is to assign aggregated weights based on the training loss of each client (the smaller the loss the greater the aggregated weight). The third way is to assign aggregated weights based on the test accuracy of each client (the higher the accuracy the higher the aggregated weight).

The algorithm's default way of averaging the weights is to assign the weights based on the size of the dataset (the source code assignment method), which we used as a benchmark for the subsequent analysis. After experimentation and analysis of the data, we found that the first modified approach, which assigns weights equally to each client, has a large improvement in accuracy compared to the original approach. The second modified approach, which assigns aggregated weights based on the training loss of each client (the smaller the loss the greater the aggregated weight), has an increase in accuracy compared to the original approach, but not as much as the previous approach. The third modification, which assigns aggregation weights based on the test accuracy of each client (the larger the accuracy the larger the aggregation weights), shows a significant improvement in accuracy compared to the original approach, and is the same as the average weight assignment.

5 Conclusion

Through experiments on the pathological non-IID dataset, we evaluate the performance of the MOON algorithm and propose optimization ideas for this dataset. The MOON algorithm can be further investigated and improved in the future to accommodate a wider range of statistical heterogeneity data scenarios. All other parameters being equal, the number of local iterations in the MOON algorithm needs to be controlled within 10 to obtain high accuracy and reduce the communication overhead and weaken the impact of system heterogeneity; the total number of clients needs to be controlled within 4, after 5 there is a significant negative impact on the accuracy; the control variable μ of the MOON algorithm needs to be controlled around 0.5 to obtain The control variable μ of the moon algorithm needs to be controlled around 0.5 so that a relatively high accuracy rate can be obtained. For the adjustment of aggregation weights, among the three additional ways we added, all three ways can provide higher accuracy than the original way. MOON algorithm, as a new distributed deep learning algorithm, can effectively solve the problem of non-independent and homogeneous data distribution, which makes it potentially applicable to several fields, such as medical and financial. With hierarchical aggregation, it can effectively support larger datasets, which will help accelerate the training process and improve the efficiency of the algorithm. the MOON algorithm supports a variety of models, such as neural networks, logistic regression, convolutional neural networks, etc., which can meet the needs of practical applications. the MOON algorithm has good improvement prospects and is expected to become one of the important algorithms in the field of distributed deep learning in the future. In general, the future outlook of MOON algorithm experiments is very broad, and the experiments can be improved in many aspects in the future, such as expanding the data scale, improving the data quality, optimizing the performance of the MOON algorithm, including speeding up the algorithm, reducing the computational complexity of the algorithm and other aspects, and better verifying the effectiveness of MOON algorithm. And it can be applied to specific scenarios such as intelligent manufacturing, intelligent transportation, intelligent medical care, etc. to verify its applicability and feasibility in different fields, and to guide the selection and optimization of MOON algorithms in practical applications.

Acknowledgment

All the authors contributed equally and their names were listed in alphabetical order.

References

1. Yang, Q., Liu, Y., Chen, T., and Tong, Y.: "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10(2), pp. 1–19, 2019).
2. Li, Q., He, B., and Song, D.: "Model-contrastive federated learning." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16529–16538, 2021.
3. Zhu, H., et al.: "Federated learning on non-IID data: A survey." arXiv preprint arXiv:2106.06843 (2021).
4. Smith, V. et al. 'Federated Multi-Task Learning'. NeurIPS, pp. 4424–4434, 2017.
5. Zhao, Yue, et al. "Federated learning with non-iid data." arXiv preprint arXiv:1806.00582 ,2018.
6. Wang, Shiqiang, et al. "Adaptive federated learning in resource-constrained edge computing systems." IEEE Journal on selected areas in Communications 37.6, pp. 1205–1221, 2019.

7. Sattler, Felix, Klaus-Robert Müller, and Wojciech Samek. "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints." IEEE Transactions on neural networks and learning systems 32.8, pp. 3710–3722, 2020.
8. Fallah, Alireza, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized federated learning: A meta-learning approach." arXiv preprint arXiv:2002.07948, 2020.
9. Diao, Enmao , J. Ding , and V. Tarokh . "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients."arXiv preprint arXiv:2010.01264, 2020.
10. Nguyen, Dinh C., et al. "Federated learning for Internet of things: A comprehensive survey." IEEE Communications Surveys & Tutorials 23.3, pp. 1622–1658, 2021.
11. Qinbin Li; Bingsheng He; Dawn Song, "Model-Contrastive Federated Learning." 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10708–10717, 2021.
12. X. Zeng et al., "Feature-Contrastive Graph Federated Learning: Responsible AI in Graph Information Analysis," in IEEE Transactions on Computational Social Systems, pp. 1–11, 2022.