Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(9): 5226–5249.
DOI: 10.3934/era.2023266
Received: 19 January 2023
Revised: 27 June 2023
Accepted: 29 June 2023
Published: 19 July 2023

Research article

FedSC: A federated learning algorithm based on client-side clustering

Zhuang Wang, Renting Liu*, Jie Xu and Yusheng Fu

University of Electronic Science and Technology of China, Chengdu 611731, China

* **Correspondence:** Email: liurt@uestc.edu.cn.

Abstract: In traditional centralized machine learning frameworks, the consolidation of all data in a central data center for processing poses significant concerns related to data privacy breaches and data sharing complexities. In contrast, federated learning presents a privacy-preserving paradigm by training models on local devices, thus circumventing the need for data transfer. However, in the case of non-IID (non-independent and identically distributed) data distribution, the performance of federated learning will drop. Addressing this predicament, this study introduces the FedSC algorithm as a remedy. The FedSC algorithm initially partitions clients into clusters based on the distribution of their data types. Within each cluster, clients exhibit comparable local optimal solutions, thus facilitating the aggregation of a superior global model. Moreover, the global model trained by the previous cluster serves as the initial model parameter for subsequent clusters, enabling the incorporation of data contributions from each cluster to foster the development of an enhanced global model. Experimental results corroborate the superiority of the FedSC algorithm over alternative federated learning approaches, particularly in non-IID data distributions, thereby establishing its capacity to achieve heightened accuracy.

Keywords: machine learning; federated learning; data protection; privacy protection; non-IID

1. Introduction

Nowadays, smart terminals, such as smart bracelets, are becoming increasingly popular. These devices are easy to carry and have many powerful sensors that can detect various inputs from the wearer's body [1, 2]. These data have strong research value. Using these data to train machine learning models will hopefully make these smart terminals smarter to better serve people. As people pay increasing attention to the protection of data privacy [3, 4], it is impossible and inadvisable to collect these private data, which leads to the problem of data silos [5]. However, if only local data is used for training, then the following problems emerge: 1) insufficient local data leads to model convergence difficulties and poor performance, and 2) limited diversity of data types hampers the model's generalization capability. As a solution, federated learning is proposed [6, 7]. In federated learning, each participant uses their

own private data to train the local model that is sent to the central server for aggregation to obtain the global model. Under the framework of federated learning, participants' private data are not exported locally, which protects participants' data privacy [8–10].

Although federated learning offers significant advantages in terms of data privacy protection compared to centralized training, it also faces numerous challenges [11, 12]. Among these challenges, heterogeneity stands out as the most crucial one [13, 14]. System heterogeneity arises when participants have varying storage capacities and computing power [15, 16]. Statistical heterogeneity, on the other hand, emerges when participants exhibit different data type distributions and data volumes, leading to the non-IID (non-Independent and Identically Distributed) problem [17]. While hardware advancements have gradually addressed system heterogeneity, statistical heterogeneity remains a persistent challenge. Non-IID data are prevalent in real-life scenarios, and federated learning's performance experiences a notable decline when confronted with such data. The classical federated learning algorithm proposed by Google [18–21], FedAvg (Federated Averaging), fails to achieve superior performance across all clients compared to models trained locally by individual clients. Some participants even experience minimal benefits or inferior performance when participating in federated learning, discouraging their willingness to engage in the process. The existing FedAvg algorithm in federated learning no longer meets practical requirements, necessitating the development of a novel algorithm to address the statistical heterogeneity challenge posed by non-IID data distributions. This research aims to tackle this challenge and provide a solution in this paper.

The negative effects of non-IID data on FedAvg can be explained by the model parameters.

We define $l(x_i, y_i; w)$ as the loss of the prediction on example (x_i, y_i) made with model parameters w. We aim for:

$$\underset{w \in R^d}{Minf}(w) \ \ where \ f(w) \overset{def}{=} \frac{1}{n} \sum_{i=1}^{n} l(x_i, y_i; w) \tag{1.1}$$

In the centralized machine learning environment, let $w_t^{(c)}$ denote the weight after $t-th$ update in the centralized machine learning. Then, the model parameters are updated in the following way:

$$w_t^{(c)} = w_{t-1}^{(c)} - \eta \nabla_w f\left(w_{t-1}^{(c)}\right) \tag{1.2}$$

In the federal learning environment, we assume there are K clients over which the data is partitioned, with P_k as the set of indexes of data points on client k, with $n_k = |P_k|$. On each client, local training is conducted separately using local data:

$$w_t^{(k)} = w_{t-1}^{(k)} - \eta \nabla_w f\left(w_{t-1}^{(k)}\right) \tag{1.3}$$

Let $w_t^{(f)}$ denote the weight calculated:

$$w_t^{(f)} = \sum_{k=1}^{K} \frac{n^{(k)}}{\sum_{k=1}^{k} n^{(k)}} w_t^{(k)} \tag{1.4}$$

Finally, we define:

$$weight_{divergence} = \|w_t^{(f)} - w_t^{(c)}\| \tag{1.5}$$

The divergence between $w_t^{(k)}$,$w_t^{(f)}$ and $w_t^{(c)}$ can be visualized from the Figures 1 and 2. When data is IID, for each client k, the data distribution is almost identical to the global data distribution and

the divergence between $w_t^{(k)}$ and $w_t^{(c)}$ is small. Therefore, $w_t^{(f)}$ obtained after aggregating different $w_t^{(k)}$ according to Eq (1.4) also has very small divergence with $w_t^{(c)}$. After many iterations, $w_t^{(f)}$ is still close to $w_t^{(c)}$ and the $weight_{divergence}$ is small. At this point, federated learning can perform very well. When data is non-IID, the large difference in the distribution of data owned by each client resulting in the divergence between $w_t^{(k)}$ and $w_t^{(k)}$ becomes larger and the divergence between $w_t^{(k)}$ and $w_t^{(c)}$ also becomes larger. Therefore, the divergence between $w_t^{(f)}$ and $w_t^{(c)}$ also becomes much larger and accumulates very fast. After many iterations, the $weight_{divergence}$ becomes larger and larger. From the above analysis, it is concluded that the negative impact of non-IID data on federated learning is mainly due to the difference in data distribution of clients. Based on this, we improve the FedAvg algorithm and propose a new algorithm, which clusters clients with similar data distribution to find local IID data in the non-IID data distribution to solve the non-IID problem.

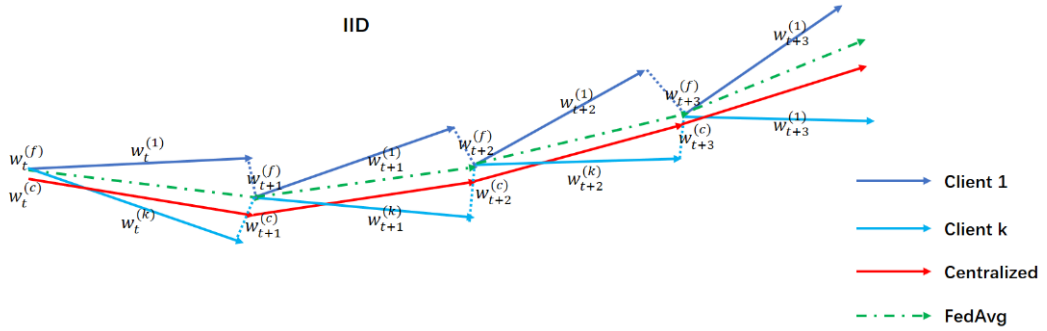

Figure 1. The weight divergence for FedAvg with IID data.

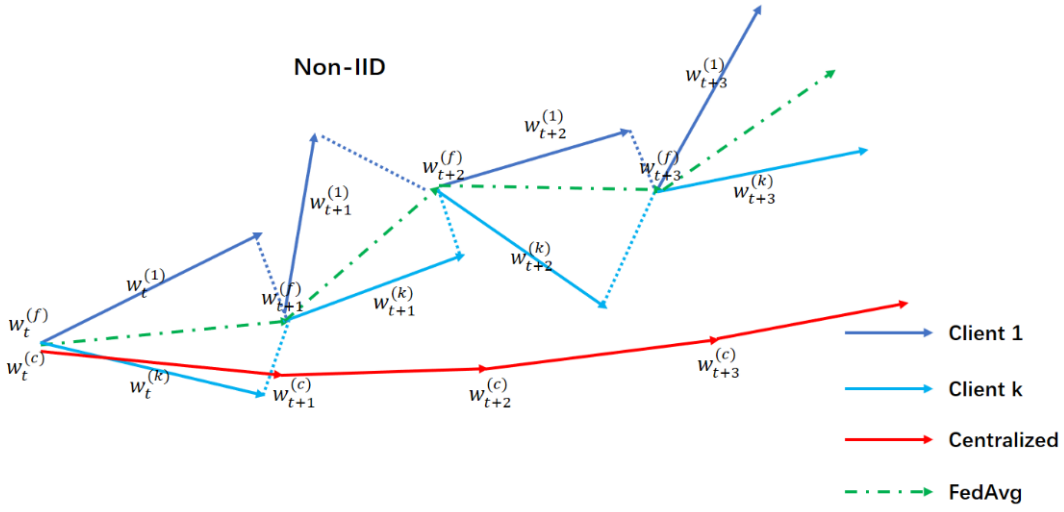

Figure 2. The weight divergence for FedAvg with non-IID data.

Contributions of this paper:

1) In this paper, FedSC algorithm is proposed to improve the accuracy of federated learning in the case of data imbalance, which helps to solve the problem of data heterogeneity in federated learning.

2) This paper reproduces a variety of federated learning algorithms, such as FedNova [22], SCAFFOLD [23], FedProx [17], etc. Through the comparisons, it demonstrates that FedSC consistently delivers strong performance across different scenarios.

3) The data training of the FedSC algorithm is performed only locally and no local data transmission is involved, which ensures data privacy and security for all participants, which helps to solve the problem of "data silos".

The structure of this paper is as follows. Section 2 presents the related work. Section 3 describes the FedSC algorithm flow in detail as well as the implementation details of each part. Section 4 presents the experimental results. Section 5 gives a summary of the full text.

2. Related work

Non-IID data distribution is common in real life [24], for example, different regions may have completely different vegetation distribution [25]. Due to data regulation and privacy concerns, meaningful real federated datasets are difficult to obtain [26]. In this paper, we use the Dirichlet distribution to simulate the non-IID distribution of data, where each client is allocated a proportion of the samples of each label according to Dirichlet distribution. Dirichlet distribution is an appropriate choice to simulate non-IID data distribution and has been used in many studies [22, 27, 28]. Specifically, we sample $p_k = Dir_N(\beta)$ and allocate a $p_{k,j}$ proportion of the instances of class k to the party j. Here, $Dir_N(\beta)$ denotes the Dirichlet distribution and β is a concentration parameter ($\beta > 0$). An advantage of this approach is that we can flexibly change the imbalance level by varying the concentration parameter β. If β is set to a smaller value, then the partition is more unbalanced.

The biggest challenge of federated learning is that its performance deteriorates when data are distributed as non-IID [29, 30]. Many methods have been put forward to solve this problem [31–34]. The FedProx algorithm is proposed in [17], which improves the local objective function based on the FedAvg algorithm. The FedProx algorithm uses an additional adjustment term in the local objective function to limit the distance between the local model and the global model so that the local model will not be too scattered. This helps to avoid local inconsistencies and improves the generalization of the model. In addition, FedProx can balance the distance between the global and local models by adjusting the regularization hyperparameters to better accommodate different data distributions. However, its shortcomings are also obvious. The client needs to carefully adjust the proportion of the adjustment item in the local objective function. If the proportion is too small, then the adjustment has little effect. If the proportion is too large, then the local update is very small and the local model converges slowly. [22] considers that heterogeneity in the clients' local datasets and computation speeds results in large variations in the number of local updates performed by each client in each communication round. Simple aggregation of such models causes the global model updates are biased. To tackle this challenge, FedNova improve FedAvg in the aggregation stage, which normalizes and scales the local updates of each party according to their number of local steps before updating the global model. Therefore, the client can iterate autonomously to make better use of the information from local data, improving the generalization capability of the model. The FedNova algorithm achieves better per-

formance on some non-IID datasets. Scaffold [23] introduces two variates (server variate and client variate), which are used to estimate the update direction of the server model and the update direction of each client. Then the difference between these two variables is used to approximate the bias in local training. Finally, the local updates are corrected by adding the difference to the local training. The Scaffold algorithm makes the federated learning model more stable, avoiding instability caused by the non-IID datasets. However, due to additional control variables, Scaffold doubles the size of each communication round compared to FedAvg. In [35], a federated learning method named FedCPF is proposed to be applied to vehicle edge computing scenarios in 6G communication networks. The method improves the efficiency and accuracy of federated learning by improving the communication method and optimizing the local training process, while reducing the communication overhead. There are also some researchers who aim to improve the distribution of non-IID data by sharing a certain percentage of the data, such as FedShare [36], Hybrid-FL [37], etc. Although the participants can be seen as trustworthy, direct data transfer still risks privacy leakage, which is against the original purpose of federated learning. In [38], the author divides clients according to the cosine similarity between the weight-updates of different clients to realize federated training of multiple models. However, it generates different personalized models for different clients rather than a unified model. In [39], TiFL was proposed, which divides participants into multiple levels according to the different performance of each participant. TiFL selects clients from the same level in every round of training in order to alleviate the discrete problem caused by the heterogeneity of resources and quantity owned by participants. The concept of client-side clustering is also employed in this paper to address the issue of non-IID data distribution. By classifying and aggregating clients based on their data distribution, the goal is to achieve a more balanced data distribution within each cluster. This approach aims to minimize the adverse effects of non-IID data on the performance of federated learning algorithms.

3. Algorithm

This section describes the FedSC algorithm in detail.

3.1. Client clustering method based on local data type distribution

Non-IID data make local model parameters divergent in the federated learning so that the central server cannot aggregate a good model; however, federated learning can give satisfactory results with IID data. Thus, we cluster the clients according to their data distribution. The clients with high data distribution similarity are divided into a cluster; hence, data distribution in this cluster is similar to IID. At this time, federated optimization within each cluster can greatly improve the performance of the model and can improve the efficiency.

Let n_i be the total amount of data samples owned by client i, n_{ij} be the number of class j data samples owned by client i, and define the data attribute of client i as $I_i = \left[n_{i1}/n_i, n_{i2}/n_i, ..., n_{ij}/n_i\right]$. Clients are clustered based on their data attributes.

A bottom-up clustering algorithm is proposed. Initially, each client is treated as an individual cluster, and clusters are progressively merged at each step. Ultimately, a cluster encompassing all samples is obtained. The detailed algorithm steps are as follows:

1) Treat each client as an individual cluster.

2) Calculate the distance between two clusters and merge the cluster with the smallest distance.

In this case, the distance between two clusters is defined as the maximum distance between any pair of clients in the two clusters. Specifically, the distance between cluster p and cluster q is defined as follows:

$$D_{p,q} = \max\left\{d_{ij} = \|I_i - I_j\|_2 \ | i \in p, j \in q\right\} \tag{3.1}$$

where d_{ij} represents the distance between the data attributes I_i of client i in cluster p and the data attributes I_j of client j in cluster q.

3) Repeat step 2 and iteratively aggregate clusters until the number of clusters meets the requirements.

In this way, the data distribution among clients within the same cluster becomes similar. Consequently, federated learning can effectively operate within each cluster, yielding improved results.

3.2. Serial training method for simulating centralized machine learning

In centralized machine learning, the model updates the model parameters by selecting a batch size of data at a time, as shown in Figure 3. All the data participate in the training process and each type of data makes its own contribution to the model fairly.

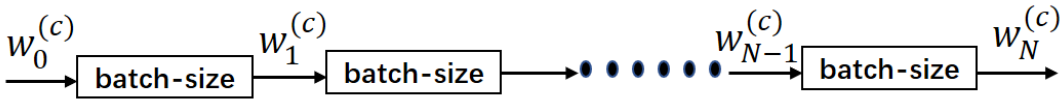

Figure 3. Training process of centralized ML.

In FedAvg algorithm, the model parameters are updated by aggregating the model parameters of each client. The central server sends the initial model parameters $w_t^{(f)}$ to the client. The client uses $w_t^{(f)}$ as the model starting point to train with the local dataset and sends the trained model $w_t^{(k)}$ to the central server. The central server aggregates the obtained models to obtain $w_{t+1}^{(f)}$. The training process of federated learning is shown in Figure 4.

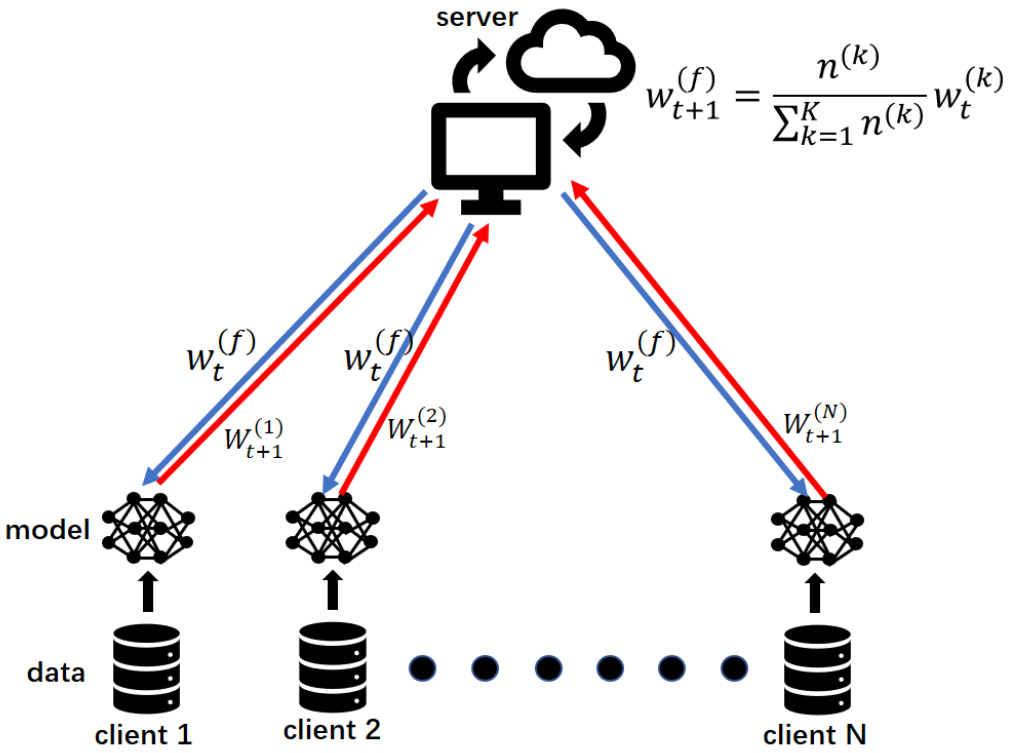

Figure 4. Training process of FedAvg algorithm.

There is a wide gap between federated learning and centralized machine learning with regard to the method of updating model parameters. Due to the clustering of clients, the traditional training process of the federated learning algorithm is no longer applicable. Therefore, we have made improvements to address this issue. We make the data in each cluster contribute to the global model of federated learning by imitating the batch processing of centralized machine learning. Treat the model aggregated by the central server from client models in a cluster as a model generated from batch-size data in centralized machine learning. Then, the central server transmits the aggregated model to the clients in another cluster for training, which corresponds to using other batch-size data to continue training the model in centralized machine learning as Figure 5.

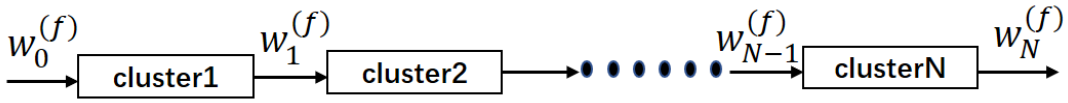

Figure 5. Training process of FedSC algorithm.

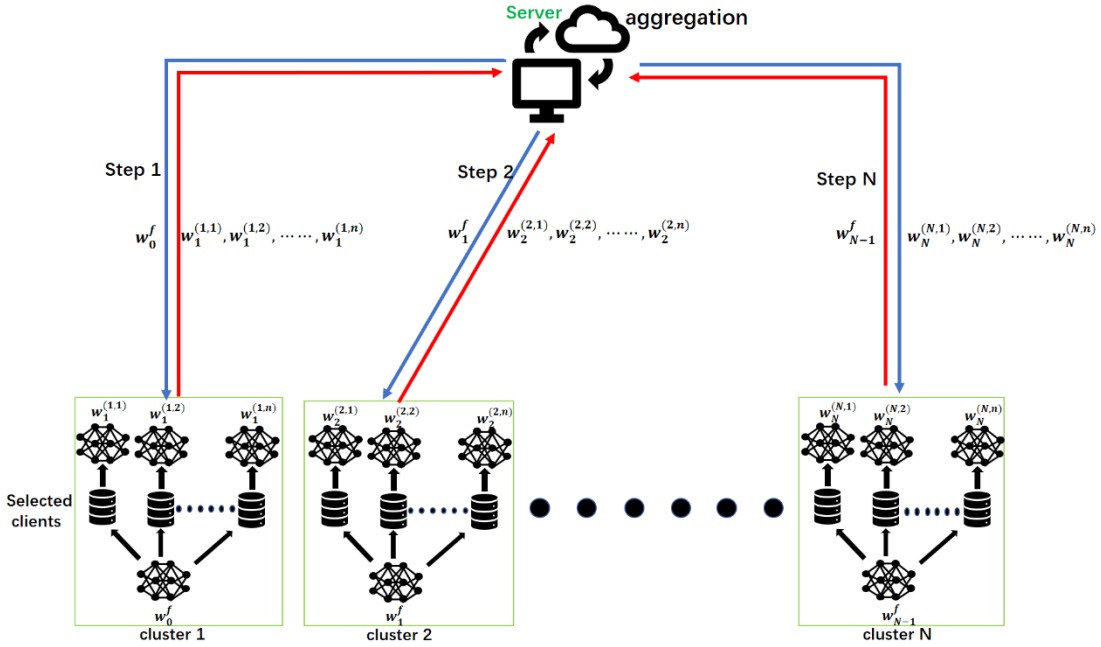

Figure 6. Flowchart of FedSC algorithm.

3.3. Detailed steps are described below

First, each client sends its data attribute I to the central server. Because the information about the proportion of the client data categories may be leaked during this process, the client encrypts the data attribute I before sending it to the central server. After receiving data attributes from all clients and decrypting them, the central server uses the clustering algorithm to aggregate all clients into N clusters. Next, the central server selects a $C - fraction$ of clients from the first cluster and send the initial global model parameters $w_0^{(f)}$ to them. After the clients train E epochs with their private data, the obtained local model parameters are uploaded to the central server. The central server performs a weighted average of the obtained model parameters according to the number of data samples of the clients to obtain the global model parameter $w_1^{(f)}$. Then, the central server selects a $C - fraction$ of clients from the second cluster and sends the global model parameters to the selected clients. The selected client takes $w_1^{(f)}$ as the initial model parameter, uses the local dataset for training E epochs, and sends the trained model parameters to the central server. After obtaining the model parameters of all clients selected from the second cluster, the central server aggregates them to obtain the global model parameter $w_2^{(f)}$. Subsequently, $w_2^{(f)}$ is sent to the clients selected from the third cluster, and so on. This process is repeated for N aggregations, enabling the original model parameter to be trained with the data from N clusters and obtain the model parameter $w_N^{(f)}$. The model parameter $w_N^{(f)}$ contains the knowledge learned from N cluster data. This constitutes a communication process. Next, the central server sends the model parameters $w_N^{(f)}$ to the clients selected from the first cluster. This process is repeated until the model converges. In centralized machine learning, all data types are traversed in batch-size units. In this paper's proposed algorithm, all data types are traversed in units of data from clients selected from each cluster. In the FedSC algorithm, data training only occurs locally on the

client side. The client and server solely exchange model parameters, without sharing any data. This approach mitigates the risk of data leakage to a certain extent and ensures the data security of clients. The schematic diagram of the FedSC algorithm is shown in Figure 6 and the pseudocode of the FedSC algorithm is shown in Algorithm 1.

Algorithm 1: FedSC algorithm

Input: Local dataset for each client
Output: FedSC algorithm
Each client sends its own data attributes I to the central server;
Server execution :
Cluster according to data attribute I of each client;
Initialize the global model parameter w_0;
for $t = 1, ..., T$ **do**
  Randomly select the participants of this round in each cluster S_t;
  $n \leftarrow \sum_{i \in S_t} |D_i|$;
  for $i \in S_t$ *in parallel* **do**
    Send the global model w_t to the selected participants;
    $\Delta w_i^t \leftarrow$ *client execution* (i, w_t);
  end
  $w_{t+1} \leftarrow w_t - \eta \sum_{i \in S_t} \frac{|D_i|}{n} \Delta w_k^t$;
end
return w_T
Client execution$(\mathbf{i}, \mathbf{w_t})$:
$w_i^t \leftarrow w_t$;
for *epoch k=1,2,...,E* **do**
  for *each batch* $b = \{x, y\}$ *of* D_i **do**
    $w_i^t \leftarrow w_i^t - \eta \nabla L\left(w_i^t; b\right)$;
  end
end
$\Delta w_i^t \leftarrow w_t - w_i^t$;
return Δw_i^t

4. Experiment

To investigate the effectiveness of FedSC algorithms on non-IID data setting, we conduct extensive experiments on three public datasets, including two image datasets (i.e., MNIST [40], FEMNIST [41]) and one tabular dataset (we collect 6000 pieces each of 10 kinds of attack flows from the network security dataset CICIDS2017 [42] as the training set, and 1000 pieces each of the corresponding attack flows as the test set). The statistics of the datasets are summarized in Table 1. We use an MLP with three hidden layers as the base model. We also use the SGD optimizer with learning rate 0.01 and the batch size is set to 64. Furthermore, we reproduce FedAvg, FedProx, Scaffold, and FedNova algorithms and run all the algorithms for the same number of rounds for fair comparison. By default, the number of rounds is set to 100, the number of clients is set to 100(P), the number of local epochs is set to 1(E),

the percentage of clients selected is set to 1(C), the β of Dirichlet distribution is set to 0.5(D), the added noise is set to 0(Z) and the number of clustering is set to 10(G), unless state otherwise. Because the selection of clients in federated learning has a certain degree of randomness, the average accuracy rate is used as the standard instead of the highest accuracy rate. The model building and simulation in this paper are implemented on Python3.7. The pytorch-gpu1.10.1 framework is mainly used to build the model.

Table 1. The statistics of datasets in the experiments.

Datasets	Training instances	Test instance	Features	Classes
MNIST	60,000	10,000	784	10
FMNIST	60,000	10,000	784	10
CICIDS2017	60,000	10,000	256	10

4.1. The influence of the order of data samples on the algorithm

The order of data samples has an important impact on machine learning. Here, we explore the influence of data sample order on FedSC algorithm. When data samples are arranged in a certain rule, machine learning learns this rule as a feature, which leads to overfitting. The clients are clustered and experimented, and then the order of the clusters is shuffled and experimented again for comparison. The experimental results are shown in Figure 7.

It can be seen from Figure 7 that the order of data samples has little influence on FedSC algorithm. In the FedSC algorithm, clients are randomly selected in each round of communication and aggregated in the cluster. These operations effectively mitigate overfitting, minimize the influence of data sample order on the model, and enhance the model's generalization ability.

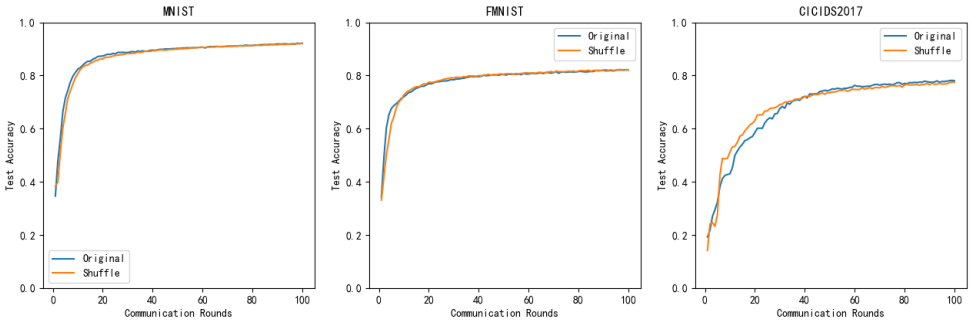

Figure 7. The influence of the order of data samples on the algorithm.

4.2. The influence of the number of cluster on the algorithm

In the FedSC algorithm, it is important to determine the optimal number of client clusters. In this experiment, the data is divided into 100 clients, and the number of clusters G is varied as 1, 10, 50, and 100. The experimental results are shown in Figure 8.

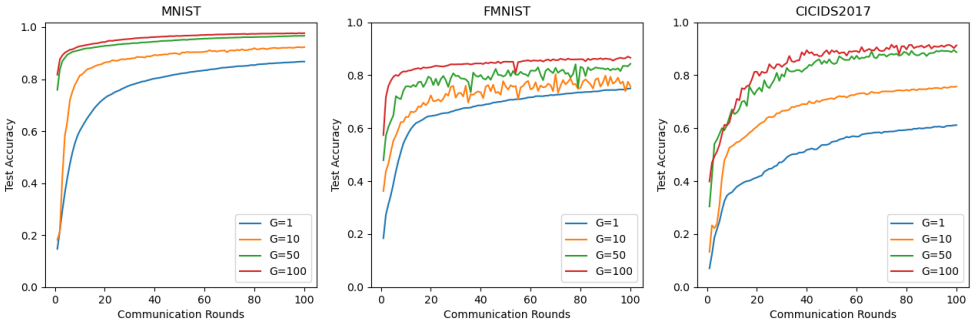

Figure 8. The influence of the number of cluster on the algorithm.

As can be seen from Figure 8, it is reasonable that the more clusters there are, the better the final result will be. In the FedSC algorithm, the model aggregated within each cluster is treated as a model trained on a batch-size of data in traditional machine learning. When the number of client clusters is small, each cluster contains a larger number of clients, resulting in scattered data distributions among clients within the cluster. This makes it challenging to aggregate a high-quality model, leading to poor performance of the final global model. When the number of clusters is large, a better model can be aggregated, but a large number of serial training consumes a long time. When G = 1, the FedSC algorithm becomes the FedAvg algorithm, which is affected by the non-IID and cannot aggregate a good global model. When G = 100, each client is equivalent to batch-size in traditional machine learning, and better results can be obtained at the cost of consuming more time. In order to balance the effect of the model and the cost of time consumed, the following experiment set the number of client clusters to 10, that is, the number of data types.

4.3. Comparison between different number of client settings

The number of clients participating in federated learning has been increasing due to the advantages it offers in addressing data uncontrollability and data leakage issues associated with traditional centralized training. Thus, we verify the effect of the number of clients on the algorithm; that is, the effect of the hyperparameter P in the algorithm on the experiment, which controls the number of clients in federated learning. The results obtained are shown in Table 2 and Figure 9.

Table 2. Average accuracy of different approaches with different number of clients.

Datasets	Setting	FedSC	FedAvg	FedNova	Scaffold	FedProx
MNIST	$p = 10$	91.44%	86.57%	87.90%	90.26%	86.51%
	$p = 100$	88.96%	78.13%	77.61%	76.20%	77.16%
	$p = 1000$	73.40%	44.21%	44.14%	47.13%	41.98%
FMNIST	$p = 10$	80.33%	77.60%	78.31%	79.45%	78.00%
	$p = 100$	65.56%	68.47%	68.34%	66.69%	67.82%
	$p = 1000$	66.65%	53.05%	49.99%	47.67%	53.16%
CICIDS2017	$p = 10$	81.20%	69.27%	69.59%	71.94%	69.38%
	$p = 100$	58.59%	49.15%	51.16%	50.28%	49.18%
	$p = 1000$	48.77%	25.48%	26.47%	25.39%	28.59%

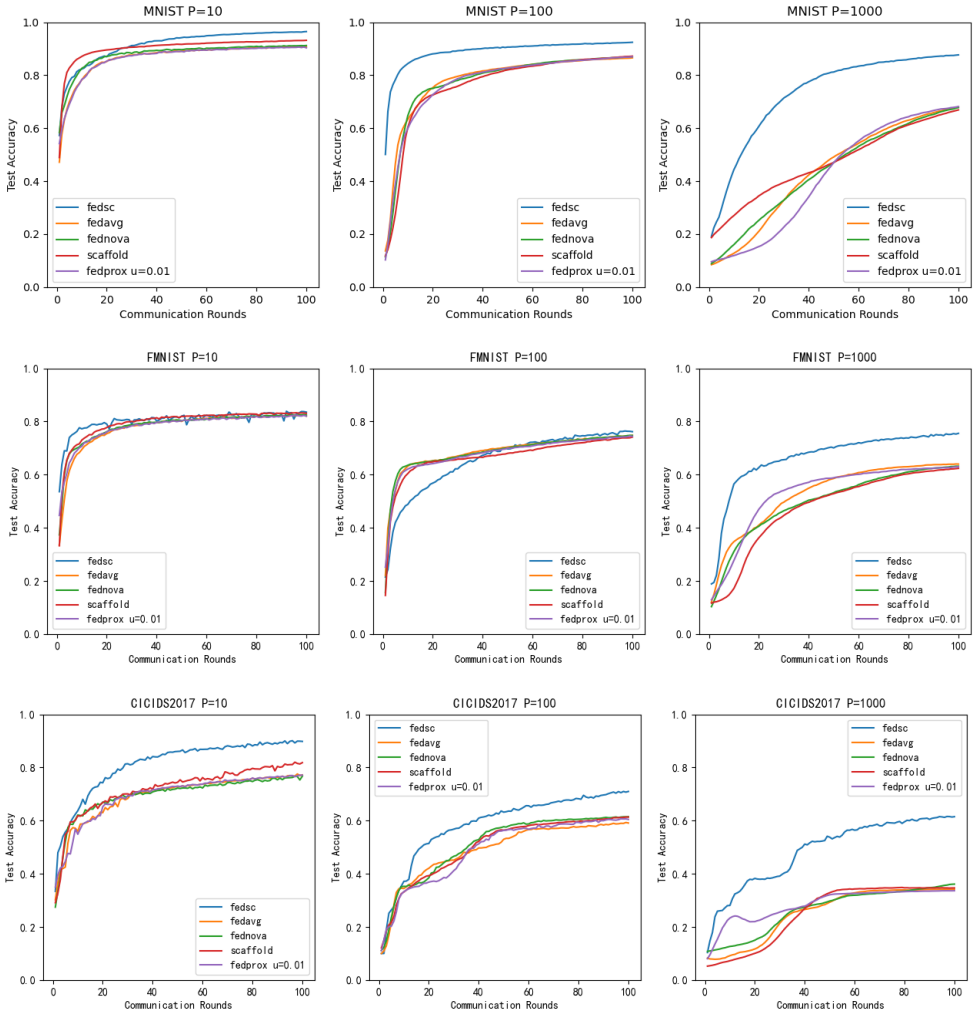

Figure 9. Training process of different approaches with different number of clients.

It can be seen from Table 2 and Figure 9 that the number of clients has a great impact on federated learning, but the FedSC algorithm is better than other algorithms. The accuracy of the FedAvg, FedNova, Scaffold and FedProx algorithms decreases significantly as the number of clients increases. For instance, when the number of clients increases from 10 to 1000, the accuracy decreases by almost half. When the total data volume is certain, the more clients there are, the less data volume each client has. Therefore, the local model trained by each client may not be accurate enough, resulting in a degradation of the performance of the global model. The accuracy of FedSC algorithm decreases as the number of clients increases, but it is still within an acceptable range. Too many clients also means more communication, which leads to increased communication overhead. Too many clients can also lead to an increase in federated learning training time, as all clients need to be trained before model aggregation can take place. Therefore, reasonable control of the number of clients is very important for the performance and efficiency of federated learning. It is important to ensure that the model can be adequately trained from diverse data while also ensuring training speed and communication efficiency.

4.4. Comparison between different data distribution settings

Now, we verify the effect of different data distributions on federated learning, i.e., the effect of the hyperparameter D on the experiments. When D becomes small, the data distribution is biased towards non-IID. When D becomes large, the data distribution is biased towards IID. The results are shown in Table 3 and Figure 10.

It can be seen from Table 3 and Figure 10 that the FedSC algorithm shows better results regardless of whether the data distribution is biased towards IID or non-IID. Additionally, all algorithms exhibit better performance when the data distribution is biased towards IID compared to when it is biased towards non-IID. In [36], it is pointed out that forcing the average fusion of model parameters with large differences will lead to a decrease in model accuracy. When the data distribution is biased towards non-IID, the dataset owned by each client are obviously different, resulting in different models trained by each client. At this time, using the FedAvg algorithm, the model parameters provided by each client vary greatly and the model after server aggregation will have significant deviations, leading to model performance degradation. Although the FedProx, FedNova and Scaffold algorithms have improved the FedAvg algorithm, the results are still less than satisfactory. In the FedSC algorithm, clients in each cluster have roughly the same data so that the local model of clients in each cluster is roughly the same. The central server aggregates these local models from each cluster, resulting in a global model that incorporates more comprehensive characteristics of such data. As a result, the FedSC algorithm achieves better and more stable performance.

Table 3. Average accuracy of different approaches with different data distributions.

Datasets	Setting	FedSC	FedAvg	FedNova	Scaffold	FedProx
MNIST	$D- > 0$	81.34%	74.17%	75.70%	78.61%	72.65%
	$D = 0.1$	84.12%	77.40%	77.70%	76.13%	76.80%
	$D- > \infty$	90.26%	76.08%	77.56%	78.54%	77.42%
FMNIST	$D- > 0$	64.18%	62.46%	62.90%	64.81%	60.58%
	$D = 0.1$	68.23%	66.60%	68.28%	67.15%	65.88%
	$D- > \infty$	80.38%	66.55%	66.43%	66.83%	66.85%
CICIDS2017	$D- > 0$	47.99%	31.70%	46.35%	47.46%	31.77%
	$D = 0.1$	51.70%	43.84%	48.70%	49.14%	42.39%
	$D- > \infty$	80.52%	56.53%	56.55%	56.64%	56.05%

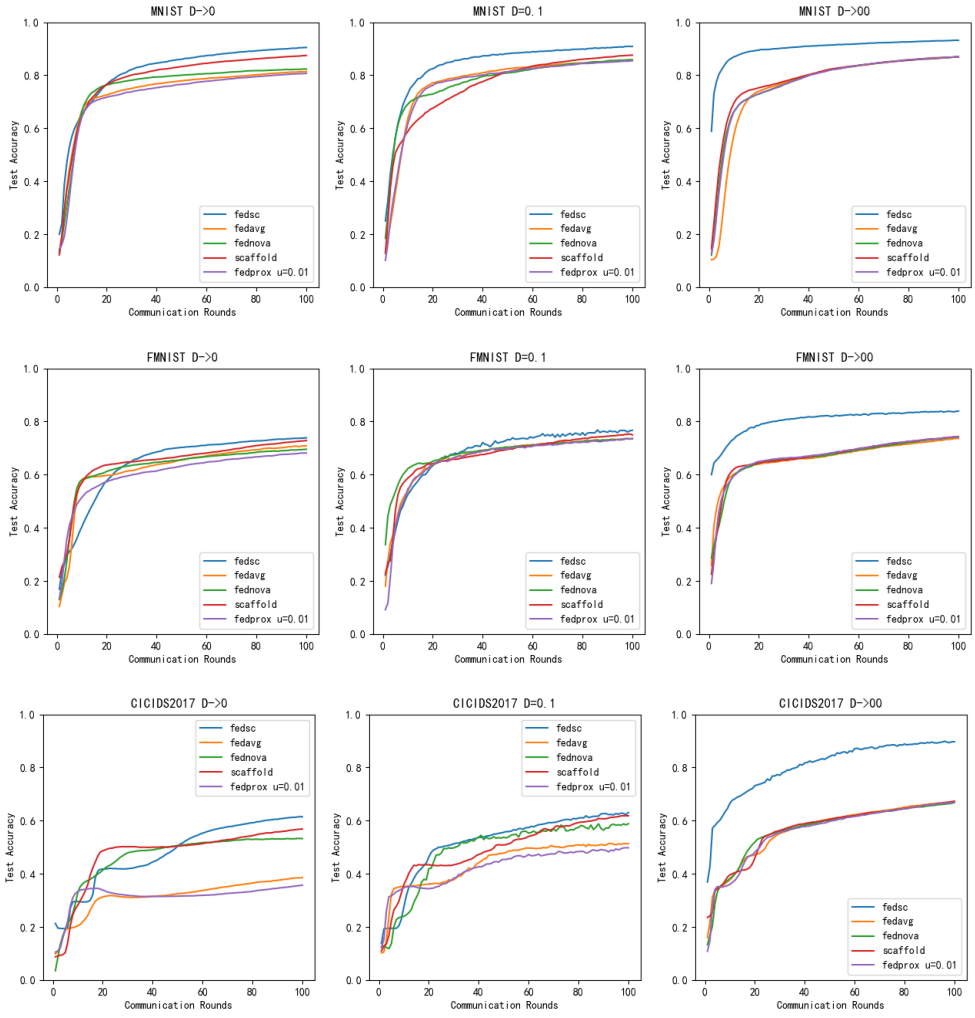

Figure 10. Training process of different approaches with different data distributions.

4.5. Comparison among different number of clients selected settings

Here, we verify the influence of the number of clients selected in each round of communication on the algorithm; that is, the influence of the hyperparameter C in the algorithm on the experiment, which controls the number of parallel clients in federated learning. Each round, C of all clients are randomly selected from all clients. The results obtained are shown in Table 4 and Figure 11.

Table 4. Average accuracy of different approaches with different number of clients selected.

Datasets	Setting	FedSC	FedAvg	FedNova	Scaffold	FedProx
MNIST	$C = 0.2$	83.39%	76.06%	75.30%	76.70%	76.70%
	$C = 0.6$	86.17%	76.01%	76.92%	77.15%	76.57%
	$C = 1.0$	86.68%	77.72%	77.48%	76.71%	77.85%
FMNIST	$C = 0.2$	73.29%	65.03%	64.63%	66.20%	65.65%
	$C = 0.6$	71.81%	67.27%	67.31%	67.16%	67.31%
	$C = 1.0$	71.84%	67.55%	65.91%	66.19%	67.06%
CICIDS2017	$C = 0.2$	65.18%	45.16%	48.24%	51.25%	43.99%
	$C = 0.6$	66.93%	48.73%	51.49%	51.99%	46.38%
	$C = 1.0$	68.14%	46.96%	51.20%	50.32%	49.76%

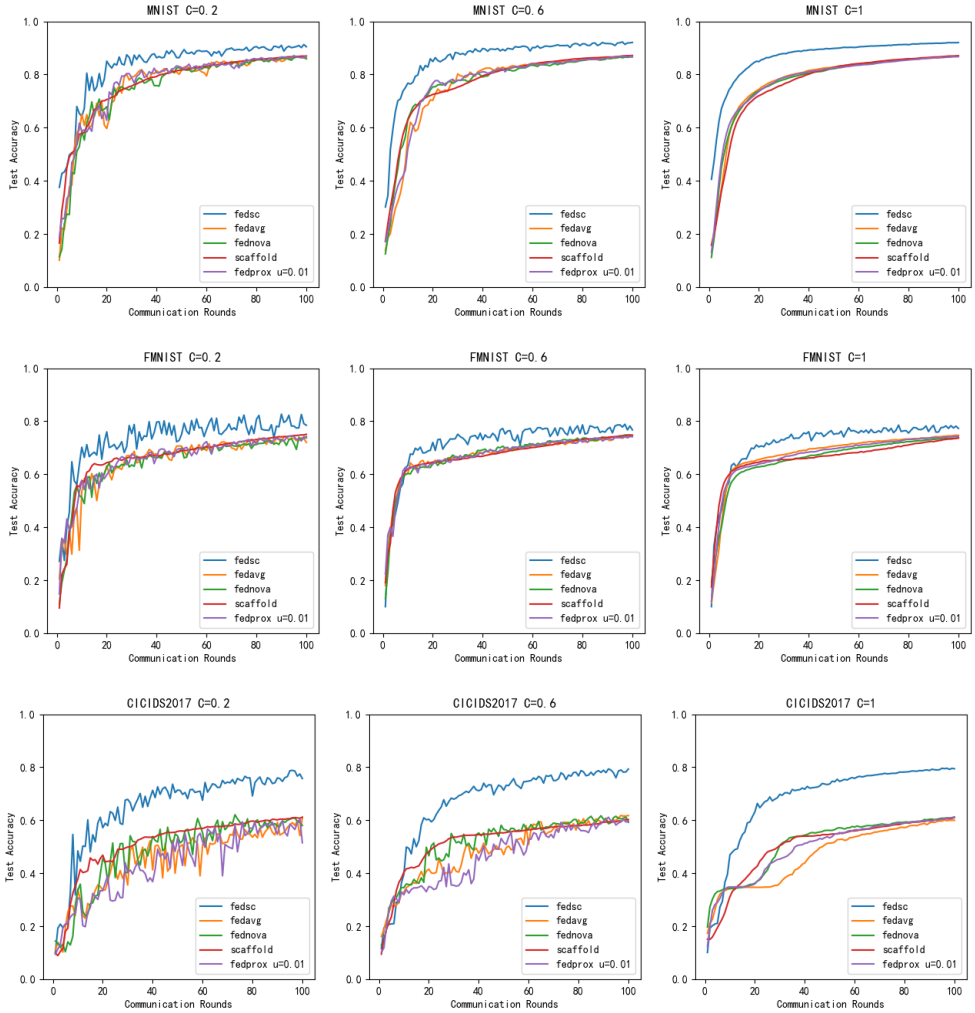

Figure 11. Training process of different approaches with different number of clients selected.

It can be seen from Table 4 and Figure 11 that the FedSC algorithm consistently achieves a higher accuracy rate compared to other algorithms in all scenarios. This superiority can be attributed to the fact that the FedSC algorithm ensures that the data within each cluster is almost IID, resulting in similar parameters among local models. Consequently, a global model with improved performance can be effectively aggregated. However, the training process of the FedSC algorithm is very unstable and the curve fluctuates a lot when fewer clients are selected each time, which is a drawback of the FedSC. The Scaffold algorithm is very stable in the training process with fewer selected clients, but its communication per round increases due to the additional variables introduced by the Scaffold algorithm.

It can also be seen that regardless of the FedSC algorithm or other algorithms, the more clients selected each time, the smaller the fluctuation of the training result, the more stable the training process, and the smoother the training curve. This is due to the fact that selecting more clients increases the number of data samples seen in each training round, resulting in more accurate results. Therefore, there will be no large fluctuations after aggregation.

Increasing the number of clients selected in each round has a positive impact on all algorithms, as it improves stability and accuracy. Selecting more clients per round of training is an effective approach to enhance performance. However, because each client needs to upload or download model parameters from the central server for each round of training, selecting more clients means more traffic. The FedSC algorithm can achieve relatively good results when C is small, which means that few clients are needed to be selected in federated learning to achieve the desired accuracy and fewer clients means a reduction in total traffic. This is helpful to solve the problem of communication bottlenecks in federated learning.

4.6. Comparison among different number of local epochs settings

Here, we verify the effect of the number of rounds updated locally by each client on federated learning, i.e., the effect of the hyperparameter E on the experiments. The results are shown in Table 5 and Figure 12.

Table 5. Average accuracy of different approaches with different number of local epochs.

Datasets	Setting	FedSC	FedAvg	FedNova	Scaffold	FedProx
MNIST	$E = 1$	86.07%	77.12%	76.47%	75.63%	78.05%
	$E = 10$	92.31%	87.50%	87.66%	89.99%	87.34%
	$E = 20$	93.46%	89.01%	89.39%	91.44%	88.93%
FMNIST	$E = 1$	77.15%	67.63%	66.45%	67.37%	67.08%
	$E = 10$	80.85%	78.79%	79.09%	80.28%	78.38%
	$E = 20$	82.58%	80.08%	80.56%	81.83%	79.91%
CICIDS2017	$E = 1$	67.77%	50.71%	50.80%	50.92%	50.52%
	$E = 10$	78.33%	68.88%	69.29%	70.80%	68.92%
	$E = 20$	80.05%	73.78%	74.06%	74.86%	73.36%

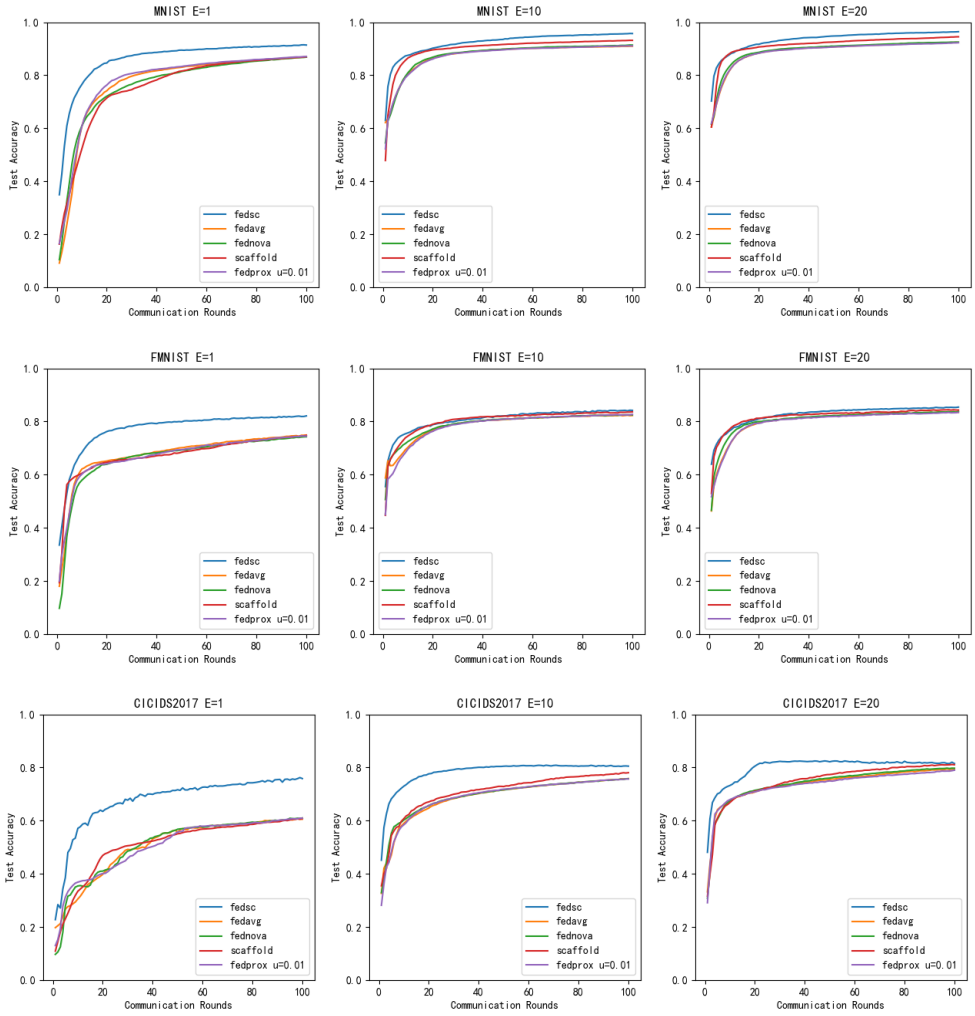

Figure 12. Average accuracy of different approaches with different number of local epochs.

As can be seen from Table 5 and Figure 12, with the change of the number of local training rounds, the FedSC algorithm keeps a strong momentum. The average accuracy of FedSC algorithm is still at a high level and the training process is relatively stable, which is sufficient to demonstrate the superiority of the proposed FedSC algorithm.

It can be seen that the increase in the number of local training rounds has a certain improvement for all algorithms. When there are fewer local training rounds, the client may not be trained sufficiently locally and the uploaded model parameters may not be accurate enough, thus affecting the performance of the global model. Increasing the number of local training rounds enables the local model to learn the features of the local data of the client better, and the effect of the global model is naturally better and more stable. Therefore, increasing the number of local training rounds is a good way to improve the accuracy and stability of the algorithm. However, a larger number of local training epochs increases the local computation of the client, which increases the training time and communication burden. Therefore, the selection of local training rounds is crucial to the performance of federated learning, and it needs to consider the factors such as data set, model complexity, computing and communication resources.

4.7. The effect of noise on federated learning

Federated learning techniques are vulnerable to Byzantine failures [43–45], biased local datasets, and poisoning attacks. Here, we verify the performance of federated learning algorithm when the client is disturbed by noise. Gaussian noise is added to the dataset to make the client training results inaccurate. Gaussian noise is popular [46], especially in images. Add Gaussian noise to each pixel point in the image data and Gaussian noise to each data in the table data. The expression of Gaussian noise is $\hat{x} \sim Gau(\mu, \sigma)$, set the mean value of Gaussian noise $u = 0$ and change the variance of Gaussian noise σ. It can be seen from Figure 13 that the larger the variance of the added Gaussian noise, the blurrier the image.

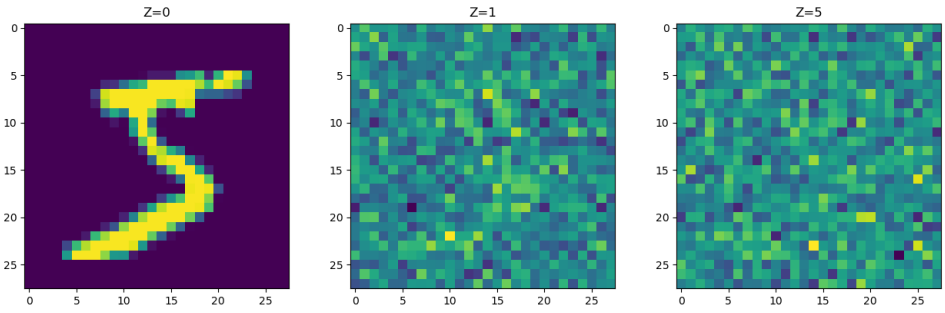

Figure 13. The effect of adding noise to the picture.

Set the variance of Gaussian noise to 0, 1 and 5 for the mnist dataset and fmnist dataset, respectively, and set the variance of Gaussian noise to 0, 1 and 2 for the CICIDS2017 dataset. The experimental results are shown in Table 6 and Figure 14.

Table 6. Average accuracy of different approaches with different noise.

Datasets	Setting	FedSC	FedAvg	FedNova	Scaffold	FedProx
MNIST	$Z = 0$	87.26%	78.61%	77.80%	77.70%	78.28%
	$Z = 1$	81.82%	73.31%	71.17%	72.39%	74.82%
	$Z = 5$	64.74%	56.56%	55.56%	53.39%	49.91%
FMNIST	$Z = 0$	74.08%	68.15%	67.85%	67.12%	67.53%
	$Z = 1$	72.50%	61.67%	61.74%	61.03%	62.38%
	$Z = 5$	44.10%	37.30%	31.67%	33.11%	29.58%
CICIDS2017	$Z = 0$	68.83%	52.79%	49.97%	46.94%	53.48%
	$Z = 1$	53.94%	44.24%	45.35%	43.71%	48.95%
	$Z = 2$	45.35%	38.93%	37.67%	35.02%	39.90%

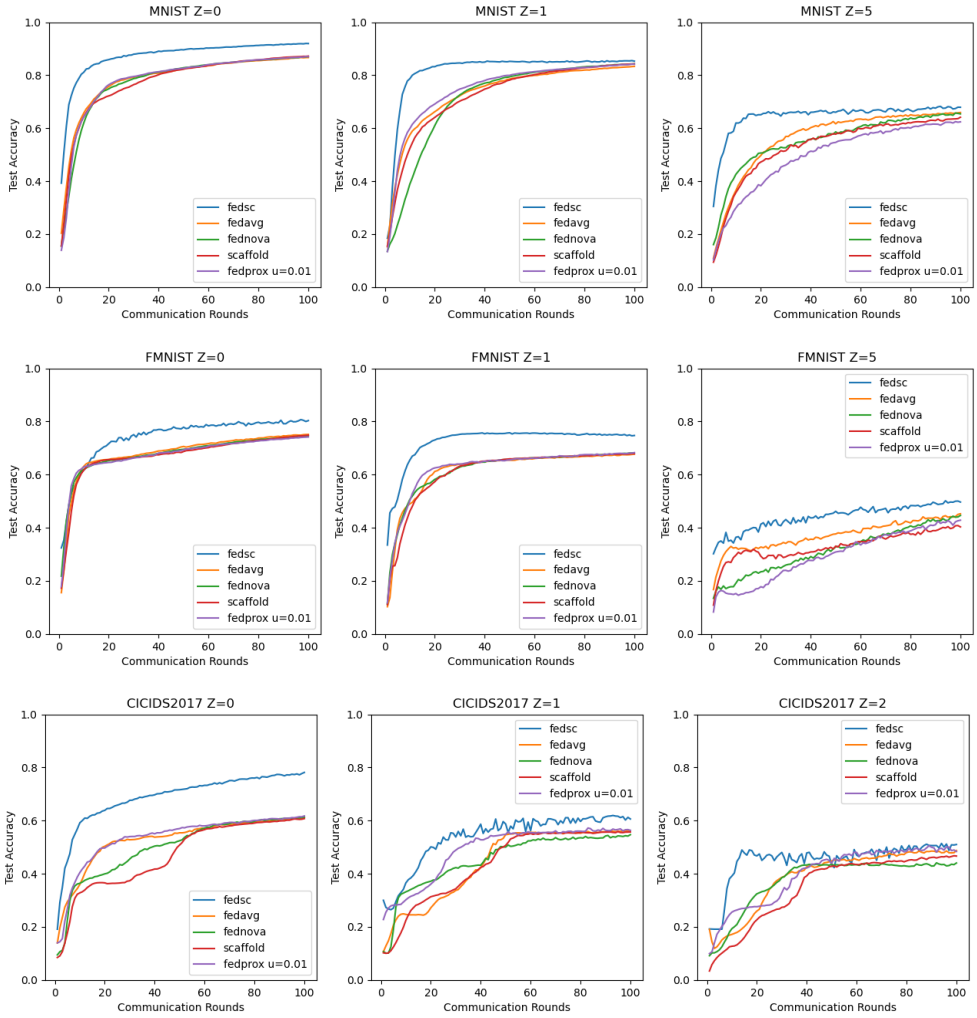

Figure 14. Average accuracy of different approaches with different noise.

The addition of noise will disturb the information in the data set, which will have a negative impact on the performance of federated learning. With the addition of Gaussian noise, the data in the dataset becomes more complex and difficult to process, making it difficult for the client to capture the features of the data during local training. Therefore, adding noise makes the training results of some clients unreliable or inaccurate, thus affecting the update results of the global model of the federated learning algorithm. As can be seen from Figure 14, the performance of all federated learning algorithms declines as the level of noise increases. Although the FedSC algorithm is also affected by noise, compared with other algorithms FedSC algorithm still gives better results. The addition of Gaussian noise can increase the privacy of data, but it will also affect the performance of the model.

4.8. Comparison of the efficiency of different federal learning algorithms

To compare the efficiency of different FL algorithms, we set P = 100, D = 0.5 C = 1, E = 1, Z = 0 for experiment. In Table 7, the total computation time of 100 communication rounds is shown, and in Table 8, the communication cost per client in each communication round is shown.

Table 7. The total computation time of 100 communication rounds.

	Mnist	Fmnist	CICIDS2017
FedSC	202 s	202 s	206 s
FedAvg	205 s	201 s	210 s
FedNova	195 s	193 s	206 s
Scaffold	264 s	268 s	272 s
FedProx	378 s	368 s	412 s

Table 8. The communication cost per client in each communication round.

	Mnist	Fmnist	CICIDS2017
FedSC	1.21 M	1.21 M	0.41 M
FedAvg	1.21 M	1.21 M	0.41 M
FedNova	1.21 M	1.21 M	0.41 M
Scaffold	2.42 M	2.42 M	0.82 M
FedProx	1.21 M	1.21 M	0.41 M

We can observe that the time consumed by the FedSC, Fedavg and FedNove algorithms is very close. However, the Scaffold algorithm introduces additional operations during training, resulting in slightly increased time consumption. FedProx directly modifies the objective, which causes additional computation overhead in the gradient descent of each batch and therefore takes the longest time. For the communication cost, the FedSC, FedAvg, FedNove and FedProx algorithms upload only their local model parameters per round, so the communication cost is the same. However, the Scaffold algorithm transmits additional control variables in each round, so its communication cost is twice that of the other algorithms.

5. Conclusions

Federated learning plays a pivotal role in the field of machine learning, particularly as the importance of data privacy continues to grow. Unfortunately, the accuracy of federated learning is significantly influenced by the distribution of data. When the data exhibits non-IID characteristics, the performance of federated learning deteriorates. This paper introduces a novel federated learning algorithm called FedSC, which outperforms other algorithms in non-IID data scenarios. The FedSC algorithm aims to identify local IID data within non-IID data to mitigate the adverse impact of non-IID data on federated learning. Through experimental comparisons, the FedSC algorithm consistently achieves higher accuracy compared to alternative federated learning algorithms. Moreover, the FedSC algorithm ensures data security by completing the model training without sharing data among participating parties. This contribution further advances the development of federated learning.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. K. Bayoumy, M. Gaber, A. Elshafeey, O. Mhaimeed, M. B. Elshazly, F. A. Marvel, et al., Smart wearable devices in cardiovascular care: where we are and how to move forward, *Nat. Rev. Cardiol.*, **18** (2021), 581–599. https://doi.org/10.1038/s41569-021-00522-7
2. M. Y. Jeng, T. M. Yeh, F. Y. Pai, Analyzing older adults' perceived values of using smart bracelets by means–end chain, *Healthcare*, **8** (2020), 494. https://doi.org/10.3390/healthcare8040494
3. Z. Lv, L. Qiao, M. S. Hossain, B. J. Choi, Analysis of using blockchain to protect the privacy of drone big data, *IEEE Network*, **35** (2021), 44–49. https://doi.org/10.1109/MNET.011.2000154
4. M. Amiri-Zarandi, R. A. Dara, E. Fraser, A survey of machine learning-based solutions to protect privacy in the internet of things, *Comput. Secur.*, **96** (2020), 101921. https://doi.org/10.1016/j.cose.2020.101921
5. Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-IID data silos: an experimental study, in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, (2022), 965–978. https://doi.org/10.1109/ICDE53745.2022.00077
6. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, B. Arcas, Communication-efficient learning of deep networks from decentralized data, in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, **54** (2017), 1273–1282. Available from: https://proceedings.mlr.press/v54/mcmahan17a.html.
7. C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, *Knowledge-Based Syst.*, **216** (2021), 106775. https://doi.org/10.1016/j.knosys.2021.106775
8. S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, et al., A hybrid approach to privacy-preserving federated learning, in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, (2019), 1–11. https://doi.org/10.1145/3338501.3357370
9. B. Yu, W. Mao, Y. Lv, C. Zhang, Y. Xie, A survey on federated learning in data mining, *Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery*, **12** (2022), e1443. https://doi.org/10.1002/widm.1443
10. S. Aich, N. K. Sinai, S. Kumar, M. Ali, H. C. Kim, M. Joo, et al., Protecting personal healthcare record using blockchain & federated learning technologies, in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, (2022), 109–112. https://doi.org/10.23919/ICACT53585.2022.9728772
11. T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, *IEEE Signal Process Mag.*, **37** (2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749

12. X. Yin, Y. Zhu, J. Hu, A comprehensive survey of privacy-preserving federated learning: a taxonomy, review, and future directions, *ACM Comput. Surv.*, **54** (2021), 1–36. https://doi.org/10.1145/3460427
13. T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, (2019), 1–7. https://doi.org/10.1109/ICC.2019.8761315
14. Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, et al., Towards taming the resource and data heterogeneity in federated learning, in *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, (2019), 19–21. Available from: https://www.usenix.org/conference/opml19/presentation/chai.
15. Y. Jiang, G. Xu, Z. Fang, S. Song, B. Li, Heterogeneous fairness algorithm based on federated learning in intelligent transportation system, *J. Comput. Methods Sci. Eng.*, **21** (2021), 1365–1373. https://doi.org/10.3233/JCM-214991
16. E. Diao, J. Ding, V. Tarokh, Heterofl: computation and communication efficient federated learning for heterogeneous clients, preprint, arXiv:2010.01264.
17. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in *Proceedings of Machine Learning and Systems*, **2** (2020), 429–450. Available from: https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.
18. H. B. Mcmahan, E. Moore, D. Ramage, B. Arcas, Federated learning of deep networks using model averaging, preprint, arXiv:1602.05629.
19. P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, et al., Advances and open problems in federated learning, *Found. Trends Mach. Learn.*, **14** (2021), 1–210. http://dx.doi.org/10.1561/2200000083
20. Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, et al., A survey on federated learning systems: vision, hype and reality for data privacy and protection, *IEEE Trans. Knowl. Data Eng.*, **35** (2021), 3347–3366. https://doi.org/10.1109/TKDE.2021.3124599
21. Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: concept and applications, *ACM Trans. Intell. Syst. Technol.*, **10** (2019), 1–19. https://doi.org/10.1145/3298981
22. J. Wang, Q. Liu, H. Liang, G. Joshi, H. V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, in *Advances in Neural Information Processing Systems*, **33** (2020), 7611–7623.
23. S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh, Scaffold: stochastic controlled averaging for on-device federated learning, in *Proceedings of the 37th International Conference on Machine Learning*, **119** (2020), 5132–5143. Available from: https://proceedings.mlr.press/v119/karimireddy20a.html.
24. Y. Esfandiari, S. Y. Tan, Z. Jiang, A. Balu, E. Herron, C. Hegde, et al., Cross-gradient aggregation for decentralized learning from non-IID data, in *Proceedings of the 38th International Conference on Machine Learning*, **139** (2021), 3036–3046. Available from: https://proceedings.mlr.press/v139/esfandiari21a.html.

25. E. O. Box, K. Fujiwara, Vegetation types and their broad-scale distribution, in *Vegetation Ecology*, (2013), 455–485. https://doi.org/10.1002/9781118452592.ch15
26. S. Hu, Y. Li, X. Liu, Q. Li, Z. Wu, B. He, The oarf benchmark suite: characterization and implications for federated learning systems, *ACM Trans. Intell. Syst. Technol.*, **13** (2022), 1–32. https://doi.org/10.1145/3510540
27. M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, Y. Khazaeni, Bayesian nonparametric federated learning of neural networks, in *Proceedings of the 36th International Conference on Machine Learning*, **97** (2019), 7252–7261. Available from: https://proceedings.mlr.press/v97/yurochkin19a.html.
28. H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, Y. Khazaeni, Federated learning with matched averaging, preprint, arXiv:2002.06440.
29. T. Hsu, H. Qi, M. Brown, Measuring the effects of non-identical data distribution for federated visual classification, preprint, arXiv:1909.06335.
30. X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-IID data, preprint, arXiv:1907.02189.
31. D. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, V. Saligrama, Federated learning based on dynamic regularization, preprint, arXiv:2111.04263.
32. Q. Li, B. He, D. Song, Model-contrastive federated learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 10713–10722.
33. X. Li, M. Jiang, X. Zhang, M. Kamp, Q. Dou, Fedbn: federated learning on non-IID features via local batch normalization, preprint, arXiv:2102.07623.
34. L. Wang, S. Xu, X. Wang, Q. Zhu, Addressing class imbalance in federated learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **35** (2021), 10165–10173. https://doi.org/10.1609/aaai.v35i11.17219
35. S. Liu, J. Yu, X. Deng, S. Wan, Fedcpf: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks, *IEEE Trans. Intell. Transp. Syst.*, **23** (2021), 1616–1629. https://doi.org/10.1109/TITS.2021.3099368
36. Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-IID data, preprint, arXiv:1806.00582.
37. N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, R. Yonetani, Hybrid-FL for wireless networks: cooperative learning mechanism using non-IID data, in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, (2020), 1–7. https://doi.org/10.1109/ICC40277.2020.9149323
38. F. Sattler, K. R. Muller, W. Samek, Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints, *IEEE Trans. Neural Networks Learn. Syst.*, **32** (2020), 3710–3722. https://doi.org/10.1109/TNNLS.2020.3015958
39. Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, et al., Tifl: a tier-based federated learning system, in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, (2020), 125–136. https://doi.org/10.1145/3369583.3392686
40. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, **86** (1998), 2278–2324. https://doi.org/10.1109/5.726791

41. H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, preprint, arXiv:1708.07747.

42. R. Panigrahi, S. Borah, A detailed analysis of cicids2017 dataset for designing intrusion detection systems, *Int. J. Eng. Technol.*, **7** (2018), 479–482.

43. L. Muñoz-González, K. T. Co, E. C. Lupu, Byzantine-robust federated machine learning through adaptive model averaging, preprint, arXiv:1909.05125.

44. P. Blanchard, E. Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: byzantine tolerant gradient descent, in *Advances in Neural Information Processing Systems*, **30** (2017). Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

45. K. Varma, Y. Zhou, N. Baracaldo, A. Anwar, Legato: a layerwise gradient aggregation algorithm for mitigating byzantine attacks in federated learning, in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, (2021), 272–277. https://doi.org/10.1109/CLOUD53861.2021.00040

46. K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: residual learning of deep cnn for image denoising, *IEEE Trans. Image Process.*, **26** (2017), 3142–3155. https://doi.org/10.1109/TIP.2017.2662206

AIMS Press