Federated Learning with Random Communication and Dynamic Aggregation

Ruolin Huang, Ting Lu, Yiyang Luo, Guohua Liu and Shan Chang

College of Computer Science and Technology,
Donghua University, Shanghai, China 201620

Abstract

Federated Learning (FL) is a setting that allows clients to train a joint global model collaboratively while keeping data locally. Due to FL has advantages of data confidential and distributed computing, interest in this area has increased. In this paper, we designed a new FL algorithm named FedRAD. Random communication and dynamic aggregation methods are proposed for FedRAD. Random communication method enables FL system use the combination of fixed communication interval and constrained variable intervals in a single task. Dynamic aggregation method reforms aggregation weights and makes weights update automately. Both methods aim to improve model performance. We evaluated two proposed methods respectively, and compared FedRAD with three algorithms on three hyperparameters. Results at CIFAR-10 demonstrate that each method has good performance, and FedRAD can achieve higher classification accuracy than state-of-the-art FL algorithms.

Keywords

Federated Learning, Random Communication, Dynamic Aggregation, Self-learning, Distributed Computing.

1. Introduction

Local devices such as mobile phones own a lot of data. However, due to data privacy and device ability, it is impractical to conduct centralized training at central server by gathering all the data from clients[1]. To address these problems, federated learning (FL)[2] is proposed. FL allows clients to train a joint global model collaboratively while keeping data locally. In this way, FL has advantage of data confidential, distributed storing and computing.

A typical FL[3] system consists of two stages connected by communication, (1) clients train local models with their local private datasets independently, and (2) server aggregates the local models into a joint global model. Since communication and aggregation are two primary performance bottlenecks of FL, interests in these two areas have increased.

In this paper, we designed a new FL algorithm named FedRAD. Random communication and dynamic aggregation methods are proposed for FedRAD. By using the combination of fixed communication interval and constrained variable intervals, random communication method enables FL system try various intervals in a single task so that we hope it can improve model accuracy. By presenting a new form of aggregation weights and making weights update automately, dynamic aggregation method enables system utilizes mutural impact between global model and local models, so as to increase task accuracy. In addition, dynamic aggregation method not only puts additional but a little computation burden on powerful server instead of resource-constrained clients, but also can apply to modern CNN. We evaluate two methods respectively, and compare FedRAD with three algorithms on three hyperparametes. Results at CIFAR-10 demonstrate that each method outperforms compared way, and FedRAD can obtain higher accuracy than state-of-the-art FL algorithms.

We organize this paper as follows. Section 2 states related works. Section 3 states two proposed methods respectively and provides an overview of FedRAD constructed by two methods, the performance of two methods and FedRAD algorithm is evaluated in section 4, section 5 summarizes the whole content of this paper.

2. RELATED WORK

Existing algorithms mainly focus on reducing the amount of parameters in communication. For example, Suresh et al.[5] uses a constant number of model in communication, and Horvath et al.[6] ignores the mantissa of parameters in model when communicating. However, these algorithms reduced the amount of parameters at the cost of decreasing task accuracy. It is know that accuracy is important to classification task such as identification task in self-driving car[7]. In order to improve model accuracy, many works can be focused on communication scheme[3]. For example, server and clients communicate once per fixed number of interval in these algorithms[5,6], which, is called fixed communication scheme. Wang et al.[8] proved that communication interval can affect the performance of the FL system. However, it is hard to get the proper communication interval before training.

Aggregation is another performane bottleneck in FL. McMahan et al.[2] proposed the standard aggregation algorithm federated averaging (FedAvg). FedAvg provides an averaging aggregation method that aggregates parameters of local models by setting weights relevant to the sizes of local datasets. Xiao et al.[9] proved that averaging parameters may not be the optimum way. In order to improve the performance of FedAvg, Sahu et al.[10] presented FedProx keeping local updates close to the original global model by adding a proximal term to the client cost functions. Although it considers the impact of global model on local updating, it increases the amount of computation on clients. To reduce the computation on clients, Yurochkin et al.[11,12] proposed Probabilistic Federated Neural Matching (PFNM) by matching the neurons of client models before averaging. However, it only works with simple architectures, e.g. fully connected network. Obviously, all of these aggregation algorithms can not consider the impact of global model on local updating, reduce the amount of computation on clients or apply to modern architecture (e.g. convolutional neural network, CNN) at the same time.

3. METHODS

In this section we introduce FedRAD. First, we state the proposed random communication method. Then, we introduce proposed dynamic aggregation method. Finally, we provide an overview of FedRAD constructed by two methods.

3.1. Random Communication

Since it is hard to use the proper communication interval before training, we first enable FL system use fixed communication scheme in the first half of training by using fixed interval. Then, system uses proposed random communication scheme in the second half of training by using constrained variable intervals. In terms of large interval leads to deterioration of the task accuracy

[8], we add a constraint on variable communication interval. Random communication method is as follows.

Let E denote total training epochs, f denote the fixed number of local training epochs, $set_t = \{t \in N^* \mid 1 \le t \le E\}$ denote the set of the training epoch, set_e denote the set of epochs communication happens, and set_{int} denote the set of communication intervals.

First, we partition set_t into three subsets according to f and the middle training epoch $\lfloor E/2 \rfloor$. In terms of f is not necessarily divisable by E , these three subsets are described as $set_{t1} = [1, \lfloor E/2f \rfloor * f]$, $set_{t2} = [\lfloor E/2f \rfloor * f, \lfloor E/f \rfloor * f]$ and $set_{t3} = [\lfloor E/f \rfloor * f, E]$ respectively.

Then, we construct set_{int} according to three subsets. For set_{t1}, we add f of quantity $\lfloor E/2f \rfloor$ into set_{int}. In this way, FL system train as fixed communication scheme. For set_{t2}, we sequentially take one subset of set_{t2} with the length of f without replacement. During each taking, we select a element randomly of taken subset and add it into set_e. After the final selection, we set the prior element of the first element in set_e as value 0, and sequentially calculate the difference value between each element in set_e with its prior element. The purpose of this way is to add a constraint on variable communication interval. i.e. By limitting the selected element (i.e, the selected communication epoch e) in the range of length f, we make the variable communication intervals the maximum value as $2f - 1$ and the minmum value as 1, so that prevent the FL system training with too much large interval. In this way, FL system trains as proposed dynaimc communication scheme. For set_{t3}, we do nothing. If f can be divisable by E , set_{t3} will not exist. Otherwise, for each $t \in set_{t3}$, the FL system training as fixed communication scheme does not communicate. So in order to compare proposed method with fixed method accurately in experiment, we do nothing on set_{t3}.

Finally, the construted set_{int} is applied to server. Server broadcasts global model together with one taken element in set_{int} to clients. The element should be taken in order without replacement. Clients then train the global model locally, setting the value of broadcast element as local training epochs.

Obviously, FL system in our method will follow fixed communication sheme when $t \in set_{t1}$, otherwise it will follow random scheme. In terms of combining two schemes to get good performance, random scheme has notable efficiency. Therefore we call this combination as random communication method. The algorithm is described as follows:

Algorithm 1 : Random Communication

```
1: Input: The total training epochs $E$, the fixed number of local training epochs $f$.
2: Initialize the 'middle' partition epoch $\lfloor E/2f \rfloor * f$, $set_e[1]=0$, $set_{int}$, $i=1$ and $j=1$
3: for $t$ in range(1, $E$) do:
4:     if $1 \le t \le \lfloor E/2f \rfloor * f$ and $t/f == 0$:
           $set_{int}[i++] = f$
5:     elif $t = \lfloor E/2f \rfloor * f + 2$:          # 'randint(a,b]' denotes selecting a integer in range of (a, b] randomly
           $set_e[++j] = randint(\lfloor E/2 \rfloor, t]$
           $set_{int}[i++] = set_e[j] - set_e[j-1] + 1$
6:     elif $t > \lfloor E/2f \rfloor * f + 2$ and $t/f == 0$:
           $set_e[++j] = randint(t-f, t]$
           $set_{int}[i++] = set_e[j] - set_e[j-1] + 1$
7: Output: $set_{int}$
```

3.2. Dynamic Aggregation

In terms of averaging is not the optimum way for aggregation [8], we proposed a new form for aggregation weights in this paper firstly. Then, by using a simple neureul network, aggregation weights can update automatically. This method is as follows.

First, we reform the aggregation weights based on fomula in FedAvg[2] as:

$$GM = \sum_{k=1}^{n} W[k] \cdot \frac{N_k}{N} \cdot M_k \tag{1}$$

Where GM denotes global model, n denotes the amount of client, N_k denotes size of local dataset D_k, N denotes size of all D_k, M_k denotes local model and $W[k]$ denotes proposed weight of local model M_k. $W[k]$ will be described initially as:

$$W[k] = \frac{acc_k^e \cdot acc_k^e}{\sum_{k=1}^{n} acc_k^e \cdot acc_k^e} \tag{2}$$

Where acc_k^e denotes task accuracy of M_k at current communication epoch e. In this way, we extend the impact of local models which have better performances.

Then, to make $W[k]$ update automatically, here we use a 2-layer neural network $NeuNet$ since neural network has the advantage of self-learning[13]. This work focused on how to set the optimization goals in $NeuNet$. In terms of $NeuNet$ and global model share the purpose of decreasing task loss, $NeuNet$ can use this shared purpose for self-learning. To that aim, $NeuNet$ requires a connection between its output layer and FL system (as shown in Figure 1). This connection is used to one-way deliver a loss value $SysLoss$ from server to output layer of $NeuNet$, to set $SysLoss$ as the loss for back propagation in $NeuNet$. The delivered $SysLoss$ is the average loss of global model after the last communication, that is, the average loss of each client tested on local test dataset before local training. In this way, server do not need to gather local private data from clients to test the loss of global model. Accordingly, let inputs of $NeuNet$ as weights of all M_k and loss as $SysLoss$, the updated aggregation weight $W[k]$ for M_k can be calculated as:

$$W[k] <= W[k] - \eta \cdot \frac{\partial SysLoss}{\partial W[k]} \qquad (3)$$

Fomula 3 is back propagation fomula in NN[14], where η denote learning rate of $NeuNet$. Thereby, dynamic aggregation method utilizes the impact of global model on local models complementarily besides considers the influence of local models on global model according to Formula 2. The algorithm is as follows.

```
Algorithm 2 : Dynamic Aggregation
1: Input: The set of accuracy on local models $SET_{ACC}$, the set of local models $SET_M$, and the set of task loss $SET_{LOSS}$.
2: Initialize 2-layer neural network model $NeuNet$, and the average task loss $Sysloss = 0$
3: for $k$ in range(1, $|SET_M|$+1) do:
4:     $W[k] = \frac{acc_k \cdot acc_k}{\sum_{k=1}^{|SET_M|n} acc_k \cdot acc_k}$
5:     $SysLoss \mathrel{+}= SET_M[k]$
6: $SysLoss = SysLoss / |SET_M|$
7: set $SysLoss$ as loss for back propagation in $NeuNet$, and get the updated $W[|SET_M|]$
7: $GM = \sum_{k=1}^{|SET_M|} W[k] \cdot \frac{N_k}{N} \cdot M_k$
8: Output: $GM$
```

3.3. FedRAD

Based on the random communication method designed in section 3.1 and the dynamic aggregation method in section 3.2, a new federated learning algorithm named FedRAD is proposed in this paper. Based on typical FL system[3], FedRAD consists of one server and several clients. The global model in server and the local model in each client adopt the same model, such as AlexNet. As can be seen from the server block in Figure 1, both methods proposed in this paper are applied to server, which has the advantage of placing the extra but small computation burden on server rather than the resource-constrained clients.

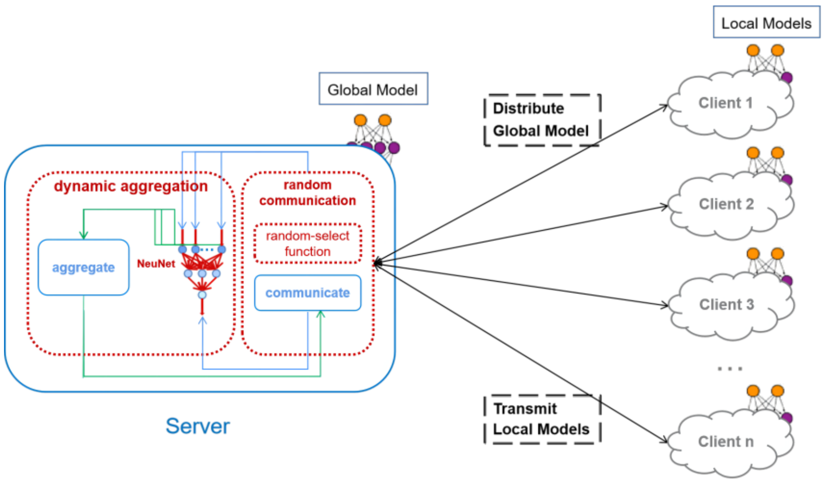

Figure 1. Structure of FedRAD

This system repeats following four steps until the training end, (1) server distributes global model and a communication interval INT generated by random communication method to clients, (2) clients first use global model to test on the local test dataset to get the loss, then train on global model using local training dataset with *INT* epochs, and test on local models to get task accuracy, (3) clients report their trained models, loss and accuracy to server, (4) server aggregates local models into a new global model according to dynamic aggregation method. The algorithm is as follows.

Algorithm 3 : FedRAD *t* is current epoch; *GM* is the global model; *Interval* is the set of communication intervals; *E* is the total of training epoch; *f* is the fixed communication interval; M_k is the local model; D_k is the local training dataset of client *k; T_k* is the local testing dataset of client *k*.

```
Server：
1：if $t$ == 1:
2：    Initialize $GM_0$
3：    $Interval$ <= Random Communication (E, f)
4：else:
5：    Receive $M_k$, $acc_k$, $loss_k$ clients report
6：    construct $SET_{ACC}$ with all $acc_k$, $SET_M$ with all $M_k$, and $SET_{LOSS}$ with all $loss_k$
7：    $GM$ <= Dynamic Aggregation ($SET_{ACC}$, $SET_M$, $SET_{LOSS}$)
8：Distributes $GM_0$ or $GM$, and $Interval[i++]$ to clients        #$i$ denotes a increment variable, which initialized as 0
Client $k$：
9：initialize $M_k$ <= $GM$, $e$ <= $Interval$
10：tests $M_k$ on $T_k$ to get $loss_k^0$
11：trains $M_k$ on $D_k$ with $e$ epochs locally to get $acc_k^{+e}$, $M_k^{+e}$
12：communicate with server to report $loss_k^0$, $acc_k^{+e}$, $M_k^{+e}$
```

4. EXPERIMENTS

In this section, we first state experiment settings in section 4.1. Then, we evaluate two proposed methods respectively and present an evaluation of FedRAD compared with three algorithms on three hyperparametes in section 4.2.

4.1. Setup

4.1.1. Task and dataset

Our training task is image classification on CIFAR-10. We separate smaller datasets of various sizes from the training set and further use data augment method to simulate several conditions. Test images are used for a global test after each round. For different models, we record the test accuracy as the metric to compare model performance.

4.1.2. Baselines

For random communication method, we compare it with typical fixed method. For dynamic aggregation method, we compared it with FedAvg and FedProx. For FedRAD, we compare it with three algorithms FedAvg, FedProx and centralized training. In addition, in order to ensure the accuracy of comparison results, modern CNN architecture MobileNet is used as learning model among all comparison algorithms.

4.2. Results

4.2.1. Performance of two methods

Random communication Since FedAvg and FedProx both communicate as fixed scheme, here we compared proposed random method with typical fixed method. Fig. 2(a) and Table 1 show the compared results, where “fixed” denotes a FL system with fixed communication method, and

"Random" denotes system with random communication method with the same communication amount T under the setting $c = *$. We can see in Fig. 2(a) that enlarging f does not always work for all conditions, which matches the conclusion made in the previous work[2]. In addition, since it is hard to know the proper number of communication interval for certain task before training, FL system with random communcation method can use various intervals in training, so that improve model performance. It can be seen in Fig. 2(a), in case of assigning different communication interval f as 4, 5, 6, and 7, random group performs better than the fixed group, which demonstrates the availability of proposed method.

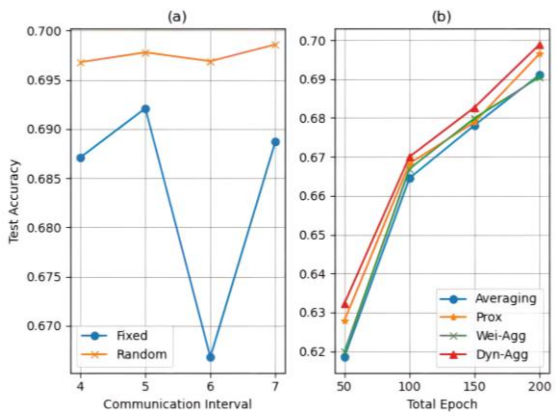

Figure 2. Comparative experiments results. (a) The performance of the model depending on the communication methods. (b) The performance of the model depending on the aggregation methods.

Dynamic aggregation Since FedAvg and FedProx have different aggregation methods, here we compared proposed dynamic method with FedAvg and FedProx. Fig. 2(b) and Table 1 show the compared results, where "Averaging" denotes a FL system with federated averaging method, "Prox" denotes system with FedProx method, "Wei-Agg" denotes system with proposed form of aggregation weights method, and "Dyn-Agg" denotes system with proposed dynamic aggregation method. It could be seen that weighted aggregation shows a little better performance than FedAvg, but it could not reach the height of FedProx. Dynamic aggregation method shows more flexible and efficient ability than others, which demonstrates the availability of proposed method.

Table 1. Trained summary on MobileNet over CIFAR-10 as shown in Figure 2.

hyperparameter	**Algorithm**	FedAvg	**Ran-Com**	FedProx	**Wei-Agg**	**Dyn-Agg**
Communication interval	4	68.71	**69.68**			
	5	69.21	**69.78**			
	6	66.68	**69.69**			
	7	68.87	**69.86**			
Iteration amount (* client amount)	5	61.85		62.79	**62.00**	**63.23**
	10	66.69		66.83	**66.69**	**67.00**
	15	67.99		67.89	**68.00**	**68.27**
	20	69.11		69.55	**69.05**	**69.89**

4.2.2. Performance of FedRAD

Dataset size It is known that model performs better when more training data is available. To simulate this scenario, we first partition the entire training CIFAR-10 dataset into n parts, where n denotes the client amount. We then augment data of original entire dataset and concatenate

n parts with data-augment parts for each client's demand. Using this strategy, we partition the training set into n sub-datasets containing 5/8/10/20 thousand (k) points each. Figure 3(a) and Table 2 show that centralized training performs better than others when dataset size is 5k, while the gap is closing as size increasing. The augment methods we used are not sufficient to fill data variety might account for the result[15]. Still it could demonstrate that FedRAD performs better than FedAvg and FedProx, and obtains comparable even slightly higher accuracy than centralized training.

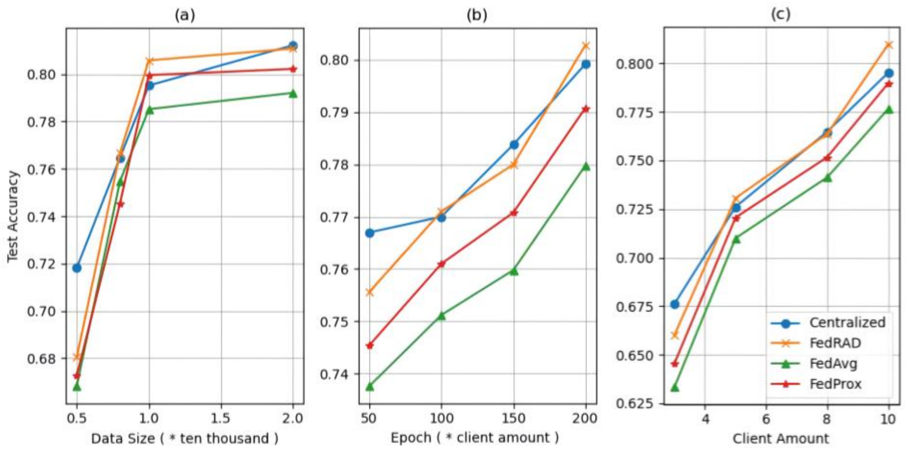

Figure 3. Comparative performances among models over three hyperparameters. (a) Influence of dataset size. (b) Influence of iteration amount. (c) Influence of client amount.

Iteration amount Training iteration also matters to model performance. Thus we further test influence of iteration amount. Results (Fig. 3(b) and Table 2) show that FedRAD obtains higher accuracy than FedAvg and FedProx, and achieves similar performance with centralized training.

Client amount We already know that model performs better as increasing the amount of training data and epoch. Challenge here is when new clients participate a FL system which already works for a while, they may not adapt to global model at short notice, which can decrease the model accuracy despite the growing data size and iteration amount. To simulate this scenario, we first let 3 clients first join the FL system, then add 2, 3, 2 clients in system respectively in each 200 epochs. Results (Fig. 3(c) and Table 2) show that FedRAD obtains higher accuracy than FedAvg and FedProx when handling new participants.

To sum up, each method we proposed outperforms typical or state-of-the-art methods. FedRAD consists of two methods obtains higher task accuracy compared with FedAvg and FedProx, and achieves similar performance with centralized training.

Table 2. Trained summary on MobileNet over CIFAR-10 as shown in Figure 3.

hyperparameter	**Algorithm**	FedAvg	FedProx	Centralized	**FedRAD**
Dataset size (* thousand)	5	66.82	67.27	68.04	**71.82**
	8	75.48	74.54	76.67	**76.48**
	10	78.52	79.96	80.58	**79.52**
	20	79.21	80.22	81.08	**81.22**
Iteration amount (* client amount)	5	73.76	74.54	75.56	**76.70**
	10	75.12	76.10	77.10	**77.00**
	15	75.98	77.08	78.00	**78.38**
	20	77.98	79.08	80.28	**79.92**

Client amount	4	63.34	64.55	66.02	**67.62**
	6	71.00	72.06	73.06	**72.61**
	8	74.13	75.17	76.37	**76.48**
	10	77.66	78.98	80.98	**79.52**

5. SUMMARY

Conclusions This paper proposed a new federated learning algorithm with random communication and dynamic aggregation (FedRAD). Random Communication method uses the combination of fixed communication interval and constrained variable intervals that FL system can try various intervals in a single task, so as to improve model performance. Dynamic aggregation method reforms aggregation weights and updates weights automately that considers mutural impact between global model and local model, aiming to increase task accuracy. Thus, FedRAD consolidates several advantages into a single framework. It considers mutural impact between global model on local model, puts additional computation burden on powerful server instead of resource-constrained clients, applies to modern architectures, and all while improves task accuracy. Though, the proposed random communication method can not address the problem of performance divergence caused by too large communication interval still as well as state-of-the-art methods, and the proposed dynamic aggregation method is a whole-wise way compared with element-wise way[11]. Thus, further works can be focused on as follows.

Future works For communication, we will try to reform communication scheme to a gradual way with introducing incremental learning (IL)[16]. Since IL has the advantage of promoting the connection of old and new tasks[17], FL system with IL can face the problem of unbalanced data distribution. And for aggregation, we will devote to design a element-wise method used for modern complex CNNs that registering neurons before aggregating[11], so as to improve model performance. Thus, we will devote to design a more effective and flexible FL algorithm than popular algorithms.

ACKNOWLEDGEMENTS

This work is supported by Shanghai Municipal Natural Science Foundation (Grant No. 21ZR1401200, Grant No.18ZR1401200), Special Fund for Innovation and Development of Shanghai Industrial Internet (Grant No. 2019-GYHLW-01004). (Corresponding author: Ting Lu)

REFERENCES

[1] Barrachina, S. , A Castelló, M Catalán, Dolz, M. F. , & Mestre, J. I. . (2021) “Pydtnn: a user-friendly and extensible framework for distributed deep learning”, The Journal of Supercomputing(4).

[2] Mcmahan, H. B. , Moore, E. , D Ramage, Hampson, S. , & Arcas, B., (2017) “Communication-efficient learning of deep networks from decentralized data”, In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pp1273-1282.

[3] Zhang, C. , Xie, Y. , Bai, H. , Yu, B. , & Gao, Y. , (2021) “A survey on federated learning”, Knowledge-Based Systems, Vol. 216, No. 1, pp106775.

[4] Peter, Kairouz. , H. Brendan, McMahan. , Brendan, Avent. , Aurélien, Bellet. , & Mehdi, Bennis. , (2019) “Advances and Open Problems in Federated Learning”, arXiv preprint, arXiv:1912.04977.

[5] Suresh, A. T. , Yu, F. X. , Kumar, S. , & Mcmahan, H. B. , (2017) “Distributed mean estimation with limited communication”, InProceedings of the 34th International Conference on Machine Learning, Vol. 70, pp3329–3337.

[6] Horvath, S. , Ho, C. Y. , Horvath, L. , Sahu, A. N. , & Richtarik, P. , (2019) “Natural compression for distributed deep learning”, arXiv preprint arXiv:1905.10988.

[7] Cao, D. , Chang, S. , Lin, Z. , Liu, G. , & Sun, D. , (2019) "Understanding Distributed Poisoning Attack in Federated Learning", IEEE, 25th International Conference on Parallel and Distributed Systems (ICPADS).

[8] Wang, H. , Yurochkin, M. , Sun, Y. , Papailiopoulos, D. , & Khazaeni, Y. , (2020) "Federated learning with matched averaging", arXiv preprint, arXiv:2002.06440.

[9] Xiao, P. , Cheng, S. , & Stankovic, V. , (2020) "Averaging Is Probably Not the Optimum Way of Aggregating Parameters in Federated Learning", Entropy, Vol. 22, No. 3, pp314.

[10] Li, T. , Sahu, A. K. , Zaheer, M. , Sanjabi, M. , Talwalkar, A. , & V Smith. , (2018) "Federated optimization in heterogeneous networks", arXiv preprint, arXiv:1812.06127.

[11] Yurochkin, M. , Agarwal, M. , Ghosh, S. , Greenewald, K. , Hoang, T. N. , & Khazaeni, Y. , (2019) "Statistical model aggregation via parameter matching", In Advances in Neural Information Processing Systems, 2019a, pp10954–10964.

[12] Yurochkin, M. , Agarwal, M. , Ghosh, S. , Greenewald, K. , Hoang, T. N. , & Khazaeni, Y. , (2019) "Bayesian nonparametric federated learning of neural networks", In International Conference on Machine Learning, 2019b, pp7252–7261.

[13] Seo, J. W. , Jung, H. G. , & Lee, S. W. , (2021) "Self-augmentation: generalizing deep networks to unseen classes for few-shot learning", Neural Networks, pp12.

[14] Zhang, D. , & Lou, S. , (2021) "The application research of neural network and bp algorithm in stock price pattern classification and prediction", Future Generation Computer Systems, Vol.115, pp872-879.

[15] Perez, L. , & Wang, J. , (2017) "The effectiveness of data augmentation in image classification using deep learning", arXiv preprint, arXiv: 1712.04621.

[16] Li, Z. , & Hoiem, D. , (2017) "Learning without forgetting", IEEE Transactions on Pattern Analysis & Machine Intelligence.

[17] Eba, B. , Ap, A. , & Ik, B. , (2021) "A comprehensive study of class incremental learning algorithms for visual tasks", Neural Networks, Vol.135, pp38-54.

AUTHORS

Ruolin Huang received her bachelor's degree in 2019, from Qufu Normal University, Rizhao, China. She is currently a master student in College of Computer Science and Technology, Donghua University, Shanghai, China. Her research interests include, Federated Learning and Computer Vision.

Ting Lu is currently an associate professor in College of Computer Science and Technology, Donghua University, Shanghai, China. Her research interests include, Wireless Network and Mobile Computing et al.

Yiyang Luo is currently a master student in College of Computer Science and Technology, Donghua University, Shanghai, China.

Guohua Liu is currently a professor in College of Computer Science and Technology, Donghua University, Shanghai, China. His research interests include, Outsourcing database and Privacy Protection et al.

Shan Chang is currently a professor in College of Computer Science and Technology, Donghua University, Shanghai, China. Her research interests include, Internet of Things and Internet of Vehicles et al.