Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/19/20231000

Optimized federated learning based on Adagrad algorithm and algorithm optimization

Enming Chu[1, †], Dengbo Li[1, †], and Yangfan Tong[2, 3, †]

[1]School of Communications and Information Engineering, Xi'an University of Posts & Telecommunications, Xian, 710121, China
[2]School of Computing and Data Science, Xiamen University Malaysia, 43900, Sepang, Malaysia

[3]CST2009143@xmu.edu.my
[†]These authors contributed equally.

Abstract. Federated learning allows you to train machine learning models without sharing your local data. Due to the No-iid problem, this paper is based on the Moon algorithm, which can have excellent performance in datasets of images with models that use deep learning and outperforms FedAvg, FedProx, and other algorithms, with the goal to decrease communication costs while enhancing efficiency more effectively. This study optimizes its gradient descent technique based on Moon's algorithm by utilizing Adaptive Gradient (AdaGrad) optimizer and combining with knowledge distillation to improve Moon's algorithm in order to better reduce communication costs and improve efficiency. That is, it reduces the loss and improves the accuracy faster and better in local training. In this paper, we experimentally show that the optimized moon can better solve the communication cost and improve the accuracy rate.

Keywords: AdaGrad, moon, federated learning, algorithm optimization.

1. Introduction

In federated learning, the problem of high interaction cost is always one of the challenges faced by federated learning. Interaction is expensive because interaction on the network is much slower than local computing due to bandwidth, energy, and so on. As a result, this study explores improving interaction effectiveness from two perspectives: minimizing interaction rounds or lowering interaction information of each round. AdaGrad is suggested as a novel optimizer method in this research, and knowledge distillation is utilized for optimizing the global model to limit the quantity of information interaction. After the introduction of AdaGrad algorithm, this paper speeds up the convergence rate of the local model based on ensuring the accuracy, and greatly improves the overall performance of federated learning compared with the SGD (Stochastic Gradient Descent) algorithm. In addition, knowledge can be transferred from the complex teacher model to the streamlined and efficient students by using the knowledge distillation method. The model enhances the global model's generalization capabilities and training speed. To produce better outcomes than the original, the AdaGrad algorithm and knowledge distilling are employed in this research to enhance the MOON method, and less communication cycles are required. The communication efficiency is greatly improved and the communication cost is reduced. The problem of high communication cost in traditional federal science is optimized.

DOI: 10.54254/2755-2721/19/20231000

2. Theoretical basis

2.1. Federated learning

Federated learning is gaining popularity in both industry and academia [1], [2]. Federated learning is an artificial intelligence configuration, also known as collaborative learning, in which several clients work together to train models beneath the supervision of a single server, while the training data is kept decentralized. It may be defined as the utilization of data stored by separate nodes to accomplish common modelling and improve the efficacy of AI models while protecting the confidentiality of information and adhering to regulatory requirements. Federated learning is based on the concepts of centralized data gathering and minimization, which can help to avoid many of the systemic privacy issues and costs associated with traditional centralized machine learning and data science methodologies [3].

As seen in Figure 1, there are four basic phases to federated learning training. In addition to local client training, model parameters are downloaded from the server's memory to the client, transferred from the client to the server, and aggregated on the server. Until the server's global model convergence occurs, this step is repeated [4].

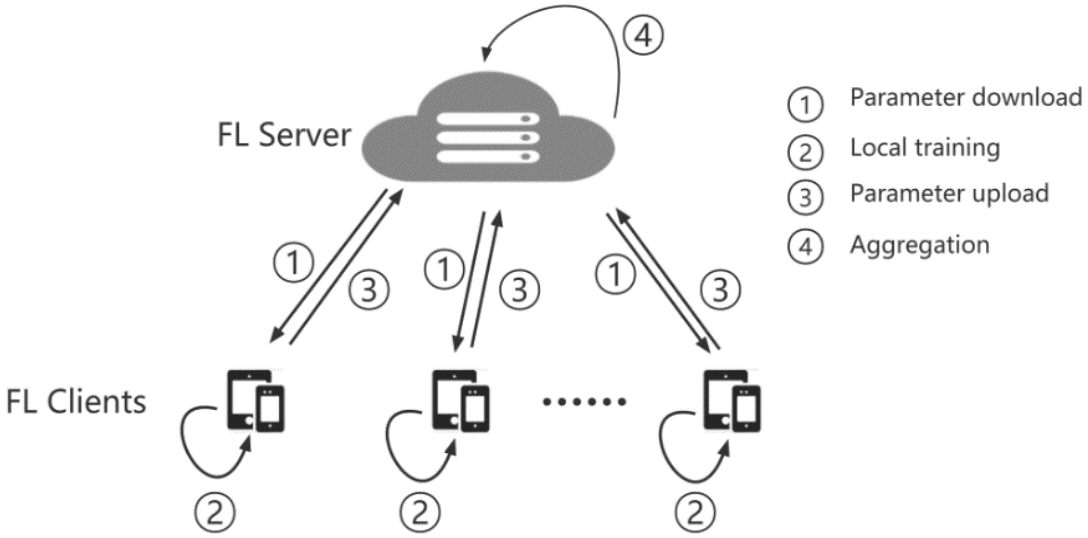


Figure1. Illustration of the four-step federal learning training process.

2.2. MOON algorithm

Without transferring local data, federated learning empowers several participants to jointly train machine learning models. To cope with the variability of local data distribution among the parties, however, is a major problem of federation learning. Although numerous studies, including FedAvg [5], have been suggested to address this issue. This study concludes that using deep learning models to provide outstanding performance in picture datasets is currently not feasible. A straightforward and efficient framework for federated learning is MOON. The main principle of MOON is to execute comparison learning at the model level by using the similarity across model representations to correct the local training of each party. Experiments have shown that the MOON algorithm significantly outperforms other federal learning algorithms, and the proposed MOON algorithm effectively solves the non-iid problem [6].

DOI: 10.54254/2755-2721/19/20231000

2.2.1. *Method.* Based on FedAvg, MOON was created as a straightforward and efficient method that only included minor, innovative alterations throughout the local training stage. In Figure 2 [6]. The FedAvg framework is displayed. MOON seeks to reduce the gap between the representations learned by the local model and the global model and to widen the gap between the depictions learned by the local model and the representations learned by the previous local model as there is always some drift in FedAvg local training and the global model learns a better representation than the local model. That example, step 2 of Figure 2 depicts how local training has improved based on the FedAvg framework.

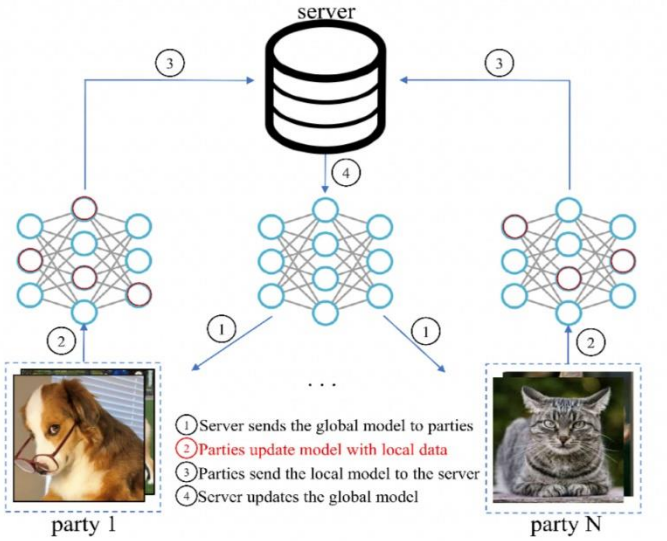


Figure 2. FedAvg framework and the four steps of FedAvg training.

A basic the encoding device (used to obtain illustration directions from the insert), the addition of a second projection head (representation mapping to a space with fixed dimensions), and an output layer (creating predictions for each class) make up the MOON network architecture. The output layer is used to provide projections for each class in the supervised setup study. The layer of output is used to provide projections for all the classes in the unsupervised setup study. For ease of representation, modelling weights w are used for convenience of representation, $Fw(\cdot)$ is used to symbolize the complete network, and $Rw(\cdot)$ is indicated by the structure of the network before the output layer (i.e., $Rw(x)$ is the mapping expression vector of input x). In local training, the local loss function is divided into two components. The first component is the standard loss term in learning under supervision. Here the SimCLR contrast learning framework is used, denoted as "sup" [7]. The suggested model contrast loss term, designated as $"con"$, is the second portion. When training locally, it first receives the global model w^t from the server side and then refreshes the model to w_i^t. The graphical representation of x is retrieved from the global model w^t (i.e. $z_{\text{glob}} = R_w^t(x)$), the image of x from the previous phase w_i^{t-1} (i.e. $z_{\text{prev}} = R_{w_i^{t-1}}(x)$), and the symbol of x from the local model that is being modified w_i^t (i.e. $z = R_{w_i^t}(x)$) for the input x. Since the global model extracts a better representation, in order to reduce the distance between z and z_{glob} and increase the distance between z and z_{prev}, this model contrast loss is defined according to NT-Xent loss as formula 1 [8].

$$\ell_{\text{con}} = -\log \frac{\exp(\text{sim}(z, z_{\text{glob}})/\tau)}{\exp(\text{sim}(z, z_{\text{glob}})/\tau) + \exp(\text{sim}(z, z_{\text{prev}})/\tau)} \tag{1}$$

where τ is the temperature coefficient and the local loss function is

$$\ell = \ell_{\text{sup}}(w_i^t; (x, y)) + \mu \ell_{\text{con}}\left(w_i^t; w_i^{t-1}; w^t; x\right) \tag{2}$$

DOI: 10.54254/2755-2721/19/20231000

where μ is the hyper parameter of the control model contrast loss weight. The local objective is minimized as

$$\min_{w_i^t} \mathbb{E}_{(x,y)\sim D^i}[\ell_{sup}(w_i^t;(x,y)) + \mu\ell_{\text{con}}(w_i^t; w_i^{t-1}; w^t; x) \quad (3)$$

2.3. Challenges

The current challenges are caused by four main aspects, system inconsistency, data inconsistency issues, privacy issues, and interaction overhead [8] (interactions on the network are much slower than local computing due to bandwidth, energy, and other issues, so it is important to improve the efficiency of interactions, and the two aspects that need to be examined rate are: reducing the number of interaction rounds or reducing the information per interaction round apparent).

3. Algorithm optimization

3.1. AdaGrad optimizer

SGD (Stochastic Gradient Descent) algorithm is one of the most used optimization algorithms in machine learning and deep learning. SGD and AdaGrad are common gradient descent algorithms used for optimizing neural networks. The main difference between them lies in how they update the weights. In SGD, the same learning rate is used to update the parameters each time, without considering the different gradient sizes of each parameter. This means that SGD may encounter situations where the gradient changes significantly, leading to an excessively large or small learning rate, which can affect the convergence speed and quality of the model.

AdaGrad, on the other hand, uses an adaptive learning rate by calculating the exponential moving average of the square of the gradient for each parameter. Specifically, in each iteration, AdaGrad calculates the cumulative sum of the square of the gradient for each parameter and uses it to scale the current gradient. As a result, the learning rate of parameters with larger gradients will be reduced, while the learning rate of parameters with smaller gradients will be increased. This allows AdaGrad to converge faster and adapt to different gradient sizes.

Therefore, the main difference between SGD and AdaGrad lies in how they update the weights. SGD uses a fixed learning rate, while AdaGrad improves performance by deceptively adjusting the learning rate. "The learning rate may have very different effects on different parameters. For those parameters that are rarely updated, this paper may want to use a relatively large learning rate, while for those that are updated frequently, this paper may want to use a relatively small learning rate [9].

The core difference between AdaGrad and SGD is the addition of the squared root of the accumulated gradient squared sum of the denominator θ_i in computing the update step size. This denominator, which can gradually become biased, will cause the update step size to decrease relatively. In the case of sparse gradients, the corresponding values in the accumulated denominator will be relatively small, resulting in a relatively large update step size. Moreover, even when the susceptibility gap is bounded, if the convergence rate is slow, the algorithm may be practically useless [10]. With the AdaGrad algorithm, this paper value its convergence speed more.

3.2. Knowledge distillation

In the experiment, this paper aim to optimize the challenge of high interaction cost in federated learning. This is one of the main current challenges of federated learning. Interaction on the network is much slower than local computation due to bandwidth, energy and other issues, so it is important to improve interaction efficiency.

In order to solve this problem, two aspects need to be considered. The first aspect is to reduce the number of rounds of information interaction. In the second aspect, this paper considers reducing the amount of information exchanged in each round. Considering the above two aspects, this paper quote Knowledge Distillation to solve the problem of high cost of information interaction.

DOI: 10.54254/2755-2721/19/20231000

The concept of Knowledge Distillation was first proposed by Hinto in his article "Distilling the Knowledge in a Neural Network"[9]. This article was originally proposed to enable lightweight networks,one of the ways to realize lightweight network is to compress the model, and knowledge distillation is one kind of model compression.

Under normal circumstances, the original model is a large complex model with redundancy, which may lead to low efficiency and poor training results if it is directly used. Therefore, this paper hope to take the required knowledge from the large model and transfer it on to the small model, and the small model can achieve the same effect as the large model, which is the main idea of knowledge distillation [11].

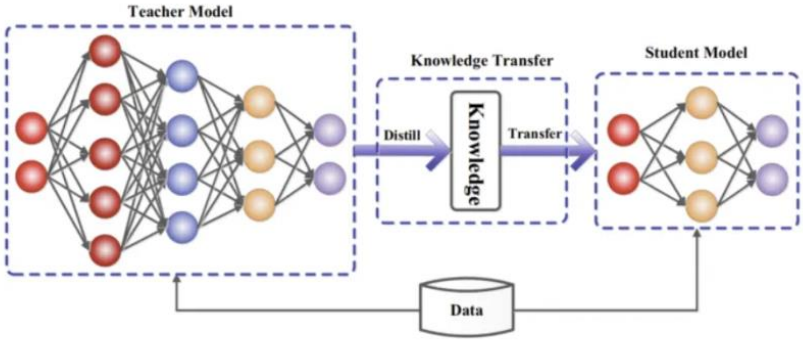


Figure 3. Flow chart of knowledge distillation.

As the figure3 show that knowledge distillation defines two models, the Teacher Model and the Student Model respectively, the Teacher Model is often a complex original model that is distilled to remove redundant knowledge and then transfer to a streamlined Student Model. In general, the use of Student Model for training has the following advantages over direct use the Teacher Model:

1. Model compression: With the original model of knowledge distilling to the smaller model, the size of the model can be reduced, thus reducing the storage and calculation costs of the model.

2. Model acceleration: The small model after distillation can achieve faster reasoning speed than the original model under the same computing resources, which is very useful for some application scenarios with high real-time requirements.

3. Improve model generalization ability: knowledge distillation can make small models learn the "soft" targets in the original model, thus improving the generalization ability and robustness of the model.

4. Model migration: By distilling the knowledge of one model into another, the training of the other model can be accelerated and its performance improved. This is useful for application scenarios such as model migration and domain adaptation.

Combining these factors, this paper consider using knowledge distillation method to optimize the global model in federated learning to reduce the high interaction cost.Specific to the application method of knowledge distillation in federated learning experiments this paper plan to use in these following aspects:

Firstly, send the predicted results of the current global model and the Teacher Model to the selected client so that the client can use this information for training of knowledge distillation. Specifically, this paper use Teacher Model to predict the current global model and send the result as a soft label to the client.

Secondly this paper use the train () method for each client participating in the training to conduct local model training,then receive the model parameters after this round of training from the client participating in the training. Finally, all model parameters after this round of training were used to update the global model by knowledge distillation. This paper expect to optimize the high interaction cost problem in these scenarios.

DOI: 10.54254/2755-2721/19/20231000

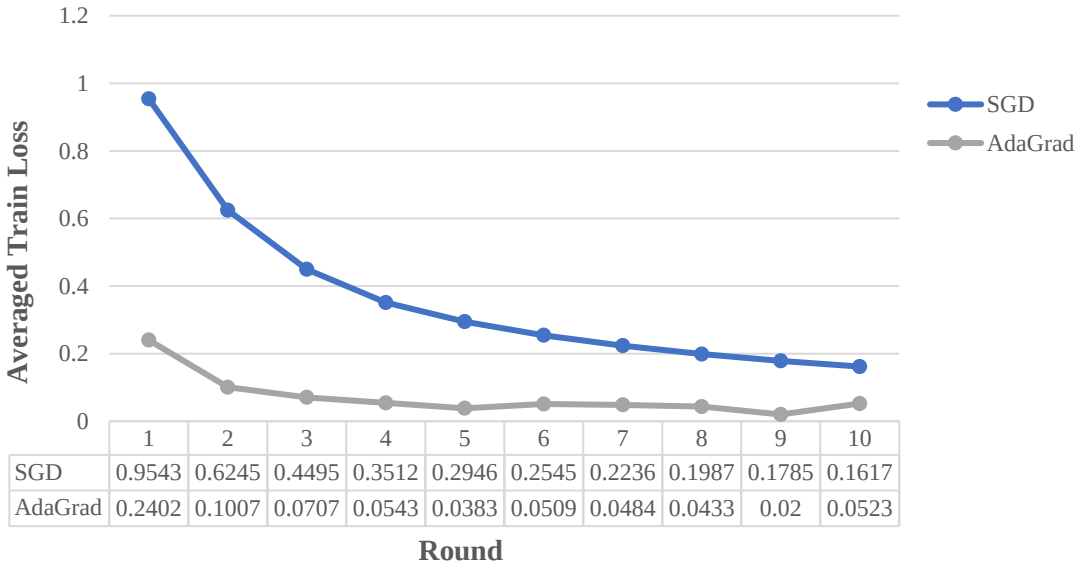
4. Simulation results and analysis

4.1. Experimental procedure

In this study, the MNIST data-set, encompassing the training and test sets, was utilized. AdaGrad and SGD algorithms were adopted as optimization algorithms for local and global training, respectively. Local training involved the usage of local data by each device for training, while in global training, all device data was pooled on a central server for training. The number of communication rounds was established at 10 rounds, implying that 10 communications between devices and servers were carried out in each training round. Loss rate and accuracy were employed as performance evaluation metrics to assess the efficacy of the algorithms during training. Trend graphs of loss rate and accuracy were generated for AdaGrad and stochastic gradient descent (SGD) algorithms in both local and global training to compare their performance in different training environments. Comparative analysis of the performance of different algorithms in local and global training was conducted, and their underlying reasons were analyzed. Based on the experimental results, it was determined that AdaGrad algorithm outperformed the SGD algorithm in both local and global training. Moreover, the use of the AdaGrad algorithm resulted in improved outcomes with fewer training rounds, thereby reducing communication overhead.

4.2. Experimental results and analysis

4.2.1. *Loss performance of SGD and AdaGrad on MINIST data-set.* The variation of loss rate on local training using SGD and AdaGrad algorithms at communication rounds of 10 respectively is shown in Figure 4.



	1	2	3	4	5	6	7	8	9	10
SGD	0.9543	0.6245	0.4495	0.3512	0.2946	0.2545	0.2236	0.1987	0.1785	0.1617
AdaGrad	0.2402	0.1007	0.0707	0.0543	0.0383	0.0509	0.0484	0.0433	0.02	0.0523

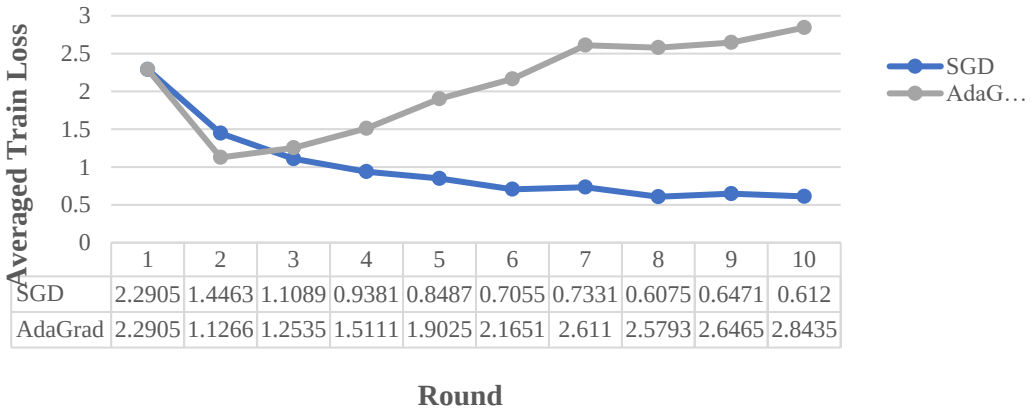
Figure 4. Trend of loss rate in local training with the number of communication rounds.

The loss rate of both SGD and AdaGrad algorithms gradually decreases as the number of communication rounds increases, and it can be concluded that the loss rate of AdaGrad is significantly lower than that of SGD at the specified communication rounds. When the first round of training was performed the AdaGrad algorithm was able to reduce its loss rate well to 0.2402 while the loss rate of the SGD algorithm only reached 0.9643. The loss rate of the AdaGrad algorithm decreased by 75 percentage points compared to the SGD algorithm. This is because AdaGrad is computationally more efficient than the SGD gradient descent algorithm since it only considers the gradient of one sample for each parameter update. With the increase of communication rounds, the loss rate of AdaGrad gradually levelled off at the communication round of 5 and finally floated around 0.0385. The loss rate of SGD

DOI: 10.54254/2755-2721/19/20231000

reached a minimum of 0.1617 at the communication round of 10, but the loss rate was still higher than that of AdaGrad by 67 percentage points, which was significantly higher than that of the SGD algorithm. This is because the AdaGrad gradient descent algorithm is an adaptive learning rate optimization algorithm, which aims to adjust the learning rate based on the historical gradient values of each parameter in order to better update the model parameters. In local training, the loss rate of local training will be low because the model can be better adapted to the data due to the smaller amount of data, while the model may be over-fitted due to the smaller amount of data, resulting in a volatile loss rate.

The performance of AdaGrad and SGD algorithms in global training is shown in Figure 5.

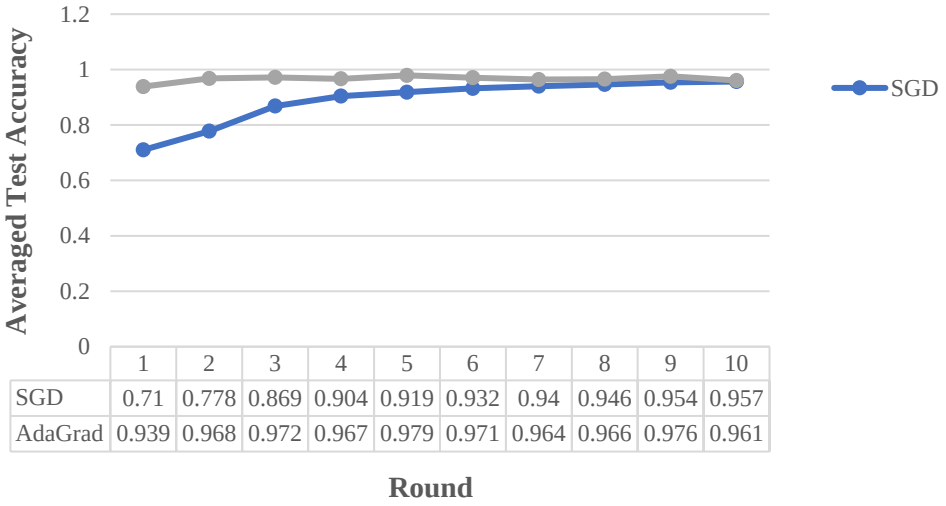


	1	2	3	4	5	6	7	8	9	10
SGD	2.2905	1.4463	1.1089	0.9381	0.8487	0.7055	0.7331	0.6075	0.6471	0.612
AdaGrad	2.2905	1.1266	1.2535	1.5111	1.9025	2.1651	2.611	2.5793	2.6465	2.8435

Figure 5. Trend of loss rate in global training with the number of communication rounds.

The loss rate of the AdaGrad algorithm is better than that of SGD only in the first two communication rounds and reaches a minimum of 1.1266 at round 2. However, the loss rate of the AdaGrad algorithm changes abruptly and gradually becomes larger as the number of communication rounds increases, while the loss rate of the SGD algorithm gradually decreases. This is because the AdaGrad is suitable for small data sets, and the loss rate becomes higher due to the large amount of data in global training, and the overall learning rate becomes smaller and smaller as the algorithm continues to iterate, which also leads to a higher loss rate in global training.

4.2.2. *Accuracy performance of SGD and AdaGrad on MINIST data-set.* The variation of accuracy rate on local training using SGD and AdaGard algorithms at communication rounds of 10 respectively is shown in Figure 6.



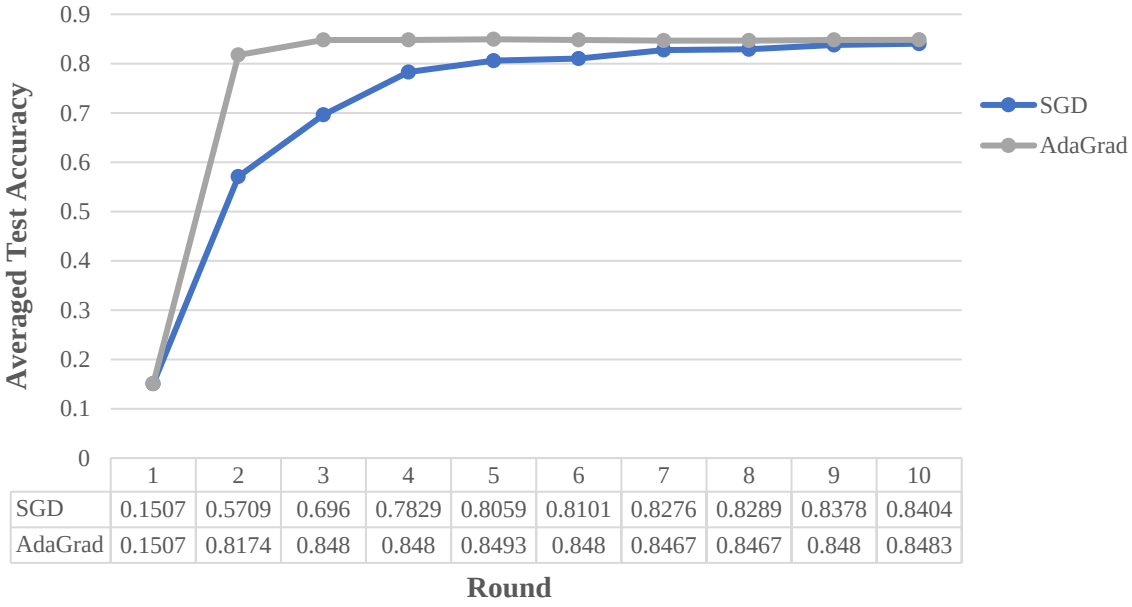
	1	2	3	4	5	6	7	8	9	10
SGD	0.71	0.778	0.869	0.904	0.919	0.932	0.94	0.946	0.954	0.957
AdaGrad	0.939	0.968	0.972	0.967	0.979	0.971	0.964	0.966	0.976	0.961

Figure 6. Trend of accuracy rate in local training with the number of communication rounds.

DOI: 10.54254/2755-2721/19/20231000

The accuracy rate of SGD and AdaGard algorithms gradually increases and tends to be stable with the increase of the number of communication rounds. The accuracy of AdaGard algorithm is significantly higher than that of SGD algorithm in the specified number of communication rounds. However, the accuracy of AdaGrad algorithm reached 0.9387 after the first round of training, while the SGD algorithm was only 0.7101. In addition, AdaGrad algorithm reached a stable value of 0.9681 after the second round of training, but SGD algorithm tended to be stable after the fifth round of training, and compared with SGD algorithm, the accuracy of AdaGrad algorithm to get stable was higher in the local model.

The variation of accuracy rate on global training using SGD and AdaGrad algorithms at communication rounds of 10 respectively is shown in Figure 7.



	1	2	3	4	5	6	7	8	9	10
SGD	0.1507	0.5709	0.696	0.7829	0.8059	0.8101	0.8276	0.8289	0.8378	0.8404
AdaGrad	0.1507	0.8174	0.848	0.848	0.8493	0.848	0.8467	0.8467	0.848	0.8483

Figure 7. Trend of accuracy rate in global training with the number of communication rounds.

The AdaGrad algorithm has a good performance in the accuracy of global model training.In the second round of global model training using AdaGrad algorithm, the accuracy reached 0.8174, while the SGD algorithm was only 0.5709, 0.2465 lower than AdaGrad. And in the subsequent training, AdaGrad algorithm reached the stable accuracy of 0.848 in the third round, but SGD algorithm reached the stable accuracy of 0.8378 after the ninth round of training.

Therefore, the AdaGard algorithm is undoubtedly batter than the SGD algorithm either in the training rounds with stable accuracy or the final accuracy rate, and it is universal in both the local model and the global model.

It can be seen from the above experiments that using AdaGrad algorithm can achieve better results with fewer training rounds, which is very helpful for us to reduce the interaction overhead.

5. Conclusion

Federated learning is a distributed machine learning approach that allows multiple clients to train a shared model without exchanging their data. However, the performance of federated learning is significantly impacted by the efficiency of network communication. Therefore, it is crucial to improve communication efficiency by reducing the number of communication rounds or minimizing the amount of information exchanged in each round. The AdaGrad algorithm is one of the optimization methods that can be utilized to accelerate the convergence rate of the local models and improve the overall performance of federated learning. Moreover, the introduction of knowledge distillation can further enhance the global model's generalization ability and training speed by transferring knowledge from a

DOI: 10.54254/2755-2721/19/20231000

larger teacher model to a smaller student model. In this study, we not only present the basic principles of federated learning but also propose two novel algorithms that aim to enhance its performance. Our experimental results demonstrate that the combined algorithm outperforms the original federated learning algorithm in terms of both efficiency and accuracy. These findings suggest that our proposed approach can effectively address the challenges associated with network communication in federated learning and facilitate the widespread adoption of this approach in real-world applications.

References

[1] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities," IEEE Internet of Things Magazine, vol. 5, no. 1, pp. 24–29, Mar. 2022.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50–60, May 2020.

[3] E. by: P. Kairouz and H. B. McMahan, "Advances and Open Problems in Federated Learning," Foundations and Trends® in Machine Learning, vol. 14, no. 1, 2021.

[4] L. Fu, H. Zhang, G. Gao, H. Wang, M. Zhang, and X. Liu, "Client Selection in Federated Learning: Principles, Challenges, and Opportunities," arXiv:2211.01549 [cs], Nov. 2022, Accessed: May 02, 2023. [Online].

[5] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," arXiv:1907.02189 [cs, math, stat], Jun. 2020.

[6] Q. Li, B. He, and D. Song, "Model-Contrastive Federated Learning," Computer Vision and Pattern Recognition, Jun. 2021.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," arXiv:2002.05709 [cs, stat], Jun. 2020.

[8] K. Sohn, "Improved Deep Metric Learning with Multi-class N-pair Loss Objective," Neural Information Processing Systems, 2016.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv:1503.02531 [cs, stat], Mar. 2015.

[10] DuchiJohn, HazanElad, and SingerYoram, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research, Jul. 2011.

[11] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized Knowledge Transfer for Personalized Federated Learning," arXiv:2111.02862 [cs], Nov. 2021, Accessed: May 02, 2023. [Online].