A Deep Gradient Compression Model Based on Tensor-Train Decomposition for Federated Learning

Xuyong Qiu, Dong Lin, Xinxin Feng, Youjia Chen, Jinsong Hu and Haifeng Zheng
College of Physics and Information Engineering, Fuzhou University, Fuzhou, China
Email: qxy15659797278@163.com, lindong@fzu.edu.cn, zhenghf@fzu.edu.cn

2020 Cross Strait Radio Science & Wireless Technology Conference (CSRSWTC) | 978-1-7281-8181-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/CSRSWTC50769.2020.9372509

***Abstract*—Federated Learning (FL) is an emerging artificial intelligence (AI) technology where the goal is to train a high-quality centralized model. However, one of challenges that FL faces is that it requires a large communication cost since it needs to exchange model parameters at each model update. In this paper, we propose a gradient compression algorithm with Tensor-Train (TT) decomposition to reduce the communication cost, where it uses low-dimensional matrices or tensors to represent high-dimensional weight parameters. Simulation results show that our method can obtain a higher compression ratio while still achieving better accuracy compared with the other compression algorithms for FL.**

***Index Terms*—Federated Learning, Gradient compression, Tensor-Train decomposition,**

I. Introduction

In recent years, with the rapid development of artificial intelligence (AI) and the Internet of Things (IoT), consumers and application providers collect data from nodes in the IoT through network edge devices (such as mobile phones, sensors, etc). These data are often transmitted to the server to perform data analysis. However, due to the privacy sensitive, it is usually not allowed to send data to the server. To solve this problem, the concept of Federated Learning (FL) was proposed by Google [1]. FL allows users to train machine learning model on local data without sending raw data to the server. However, the application of FL is largely constrained by its huge communication cost since it needs model parameters exchange between clients and server. Therefore, one of the main problems of FL is how to reduce the communication bandwidth.

Previous works show that pruning is effective to reduce network complexity by removing parameters that are not important to the performance from the model. Han et al. proposed a method based on pruning, quantify, and Huffman coding for model compression [2]. However, Liu et al. demonstrated that pruning is not always efficient, and it can be useful as a network architecture search algorithm [3]. Zhang et al. proposed a deep computation model based on CANDECOMP/PARAFAC (CP) decomposition to compress the convolutional neural network to speed up training [4]. However, the optimal rank of CP decomposition is unstable and can not be widely used in high-order tensors [5]. Zhang et al. proposed a Tensor-Train (TT) decomposition based approach for big data feature learning [6]. Szegedy et al. decomposed a 3×3 convolution kernel into two 1×1 convolution kernel to speed up training [7]. Compared with AlexNet, SqueezeNet adopts a 1×1 convolution kernel instead of a 3×3 convolution kernel, which reduces the parameters by 50 times [8]. However, these works only perform model compression on a single node and have not been applied to FL. In FL, McMahan et al. proposed to reduce the number of communications by increasing parallelism and the calculation of each client [1]. Huang et al. adopt modifying the training algorithm to increase convergence speed to decrease communication cost [9]. Yao et al. also proposed increased computation on each client by adopt a two-stream model commonly used in transfer learning and domain adaption [10]. Konecny et al. proposed a low-rank and random mask method to reduce the parameters [11], but could not get the optimal rank. Caldas et al. proposed a federated dropout mechanism to reduce communication costs [12]. Tao et al. proposed select only a small fraction of important gradients to be communicated to the FL server for parameter update [13].

In FL, communication cost is a key problem since it needs to exchange a large number of model parameters between edge devices. In this paper, we propose a gradient compression method based on TT decomposition for FL. Firstly, a model is sent from the server to the edge clients as their initial local models. Secondly, the local models are trained by using the local dataset and updated with compressed gradient with TT tensor format. Finally, the local models are sent back to the server and aggregated as a global model. The main contributions in this paper are summarized as follows:

- We propose a model gradient compression method based on TT decomposition for FL. Specifically, we use TT tensor format to represent the gradient in each model update to reduce transmission cost.
- We propose an updating approach with a back-propagation algorithm for model training at edge nodes. The proposed method can directly compute the gradients with TT format to reduce the memory requirement of training for edge nodes.
- We carry out extensive simulations on real-world dataset by considering different data distributions on edge devices. Simulation results show that the proposed method can achieve higher compression ratio while still obtaining better accuracy comparing with other methods.

The remaining of this paper is organized as follows. In Section II, we propose a model based on TT decomposition for FL. In Section III, we carry out simulations and compare performance with other methods. We conclude the paper in Section IV.

II. The Proposed Gradient Compression Model with Tensor-Train Decomposition for Federated learning

A. Tensor-Train Decomposition

Tensor-Train decomposition is a tensor factorization model that can handle tensors of any dimension. Assuming a d-dimensional tensor $\mathcal{A} \in \mathbb{R}^{p_1 \times p_2 \times \ldots \times p_d}$, it can be factorized in form of :

$$\hat{\mathcal{A}}(l_1, l_2, \ldots, l_d) = \mathcal{G}_1(l_1)\mathcal{G}_2(l_2) \ldots \mathcal{G}_d(l_d). \quad (1)$$

, where

$$\mathcal{G}_k \in \mathbb{R}^{p_k \times r_{k-1} \times r_k}, l_k \in [1, p_k], \forall k \in [1, d]. \quad (2)$$

and $r_0 = r_d = 1$. In (1), the tensor $\hat{\mathcal{A}}$ with TT format is represented as a sequence of tensor multiplication, where the tensor $\{\mathcal{G}_k\}_{k=1}^{d}$ is usually called the core-tensors of TT. The complexity of TT decomposition is determined by its rank $[r_1, r_2, \ldots, r_d]$. As shown in Fig.1, the first and last cores can be seen as matrices, and the other cores are third-order tensors.

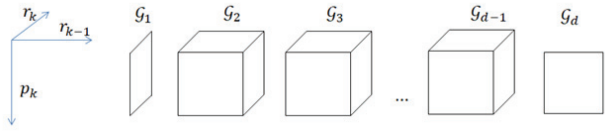

Fig. 1. Tensor-Train core. The first and last are matrices, and the others are third-order tensors.

The compression ratio of TT decomposition can be calculated as follows:

$$r = \frac{\prod_{k=1}^{d} p_k}{\sum_{k=1}^{d} p_k \times r_{k-1} \times r_k}. \quad (3)$$

B. A Gradient Compression Model for Federated Learning

The procedure of FL consists of two parts: local update and global aggregation. Firstly, each edge client i receives the model W_t from the server at time t, and then trains a new model W_{t+1}^i based on its local data. Secondly, each edge client sends back the model with TT decomposition to the server. Finally, the server receives the model from the edge clients and aggregates the models through the average algorithm to obtain a new model W_{t+1}, and send it to the edge client for a new round of iteration. The procedure is shown in Fig.2.

In this paper, we obtain the gradient Δ^i from the new model by computing:

$$\Delta^i = W_{t+1}^i - W_t. \quad (4)$$

Then, TT decomposition is used to represent the gradient Δ^i:

$$[\mathcal{G}_1(l_1), \mathcal{G}_2(l_2), \ldots, \mathcal{G}_d(l_d)] = \Delta^i. \quad (5)$$

After that, the TT cores are sent from the clients to the server:

$$\hat{\Delta}^i = [\mathcal{G}_1(l_1), \mathcal{G}_2(l_2), \ldots, \mathcal{G}_d(l_d)]. \quad (6)$$

and then a new model $\widehat{W}_{t+1}^i$ is updated for the client i.

$$\widehat{W}_{t+1}^i = W_t + \hat{\Delta}^i. \quad (7)$$

Finally, a new global model is aggregate as follows:

$$W_{t+1} = \frac{1}{n} \sum_{i=1}^{n} \widehat{W}_{t+1}^i. \quad (8)$$

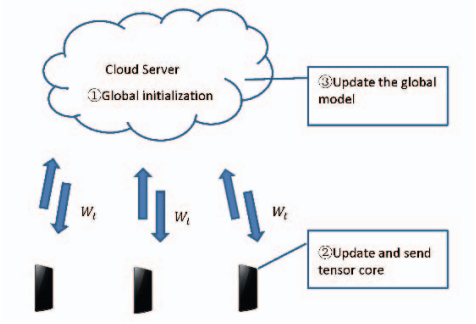

Fig. 2. Federated Learning.

III. Simulations

In this section, we adopt the STL-10 dataset to test our methods. The STL-10 dataset contains 10 types of different object pictures, and each type of object consists of 1,300 color pictures. It includes 500 training samples and 800 test samples, each with a resolution of 96×96.

In the simulations, we divide the dataset into two different data distributions: non-uniform and uniform. In the non-uniform distribution, the data samples on each sub-node do not have intersections, and the sample category on each node is a subset of all samples, denoted as IID=0; In the uniform distribution, the data samples on each sub-node do not have intersections, but each node has all sample categories, denoted as IID=1. Stochastic Gradient Descent (SGD) is used as the optimization algorithm, the learning rate is set to 0.001, and the sparsity threshold is set to 0.2. We use the synchronous model to simulate the training process of each node. The simulations are carried out on a platform with an Inter Core i5-7500 CPU and NVIDIA GeForce RTX2080GPU.

In the simulations, we adopt Convolutional Neural Networks (CNN), which contains six convolution layers and three pooling layers. The first and last three convolutional layers consist of 64 and 128 convolution kernels, respectively. The sizes of the convolutional kernels are 3×3. The fully connected layer has 1024 input and 10 output units, respectively. We assume that each node communicates with the central server after every 50 local iterations.

We compare the performance of our algorithm with the average algorithm, sparse gradient average algorithm, the average algorithm with TT model and the average algorithm with Hierarchical-Tensor (HT) model. The accuracy and parameter compression ratio are used as evaluation indexes. In Fig.3, in the case of IID=1, it can be seen that the accuracy of all algorithms is slowly approaching to the average algorithm. However, the algorithm with the TT decomposition has strong robustness and the gradient value is smaller, whose performance is close to the

original average algorithm. Therefore, it can be seen that the TT gradient compression method has certain advantages compared with the other algorithms. In Fig.4, in the case of IID=0, we can still observe that our method achieves better performance than the other algorithms and is also approximate to the original average algorithm.

TABLE.1 COMPRESSION RATIO

	rank or threshold	compression ratio
TT	4	30.68
	8	16.93
HT	4	35.19
	8	21.75
TTgradient	3	**41.42**
Sparse	0.2	5

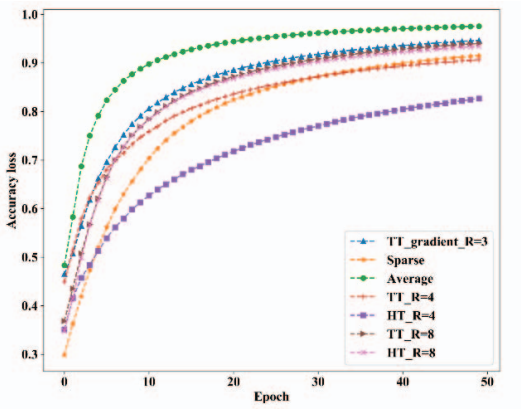

Fig. 3. Performance comparisons with different compression algorithms for IID=1.

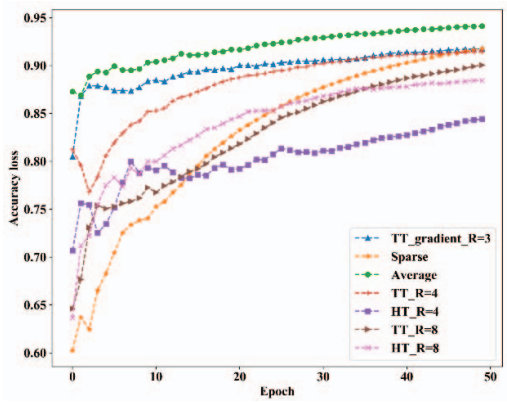

Fig. 4. Performance comparisons with different compression algorithms for IID=0.

Next, we evaluate the compression ratio for different algorithms. In this simulation, the TT rank of our gradient compression algorithm is set to 3, the threshold for the sparse gradient method is 0.2, the rank for the compression method with TT model is set to 4 or 8 and the rank for the compression method with HT model. According to the results in TABLE.1, the compression ratio of our algorithm is about 41 times, which is higher thank the other algorithms. However, from Fig. 3 and Fig. 4, we see that our algorithm can obtain better performance than other compression algorithms while still maintaining a high compression efficiency.

IV. Conclusion

In this paper, we propose a gradient compression method based on TT decomposition for FL. We find that the proposed algorithm based on gradient compression achieves better performance than the algorithm based on model compression. Simulation results demonstrate that our method outperforms the other compression algorithms in terms of accuracy while still achieve a higher parameter compression ratio.

References

[1] H. B. McMahan, E. Moore, and D. Ramage, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 1273-1282, 2017.

[2] S. Han, H. Mao and W. j. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding," *Proceedings of the International Conference on Learning Representations*, 2016.

[3] Z. Liu, M. Sun and T. Zhou, "Rethinking the value of network pruning," *Proceedings of the International Conference on Learning Representations*, 2019.

[4] Q. Zhang, L. T. Yang and Z. Chen, "An improved deep computation model based on canonical polyadic decomposition," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 10, pp. 1657-1666, 2018.

[5] X. Wang, L. T. Yang and H. Liu, "A big data-as-a-service framework: state-of-the-art and perspectives," *IEEE Transactions on Big Data*, vol. 4, no. 3, pp. 325-340, 2018.

[6] Q. Zhang, L. T. Yang and Z. Chen, "A Tensor-train deep computation model for industry informatics big data feature learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3197-3204, 2018.

[7] C. Szegedy, S. Ioffe and V. Vanhoucke, "Inception-resNet and the impact of residual connections on learning," *Proceedings of the National Conference on Artificial Intelligence*, pp. 4278-4284, 2016.

[8] B. Wu, A. Wan and F. Iandola, "Squeezedet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pp. 446-454, 2017.

[9] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost:Loss-based adaboost federated machine learning on medical data," *arXiv preprint arXiv:1811.12629*, 2018.

[10] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-4, 2018.

[11] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[12] S. Caldas, J. Konecny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[13] Z. Tao and Q. Li, "esgd: Communication efficient distributed deep learning on the edge," *In {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.