Sparse Communication for Federated Learning

Kundjanasith Thonglek*, Keichi Takahashi†, Kohei Ichikawa*, Chawanat Nakasan‡, Pattara Leelaprute§ and Hajimu Iida*
* Nara Institute of Science and Technology, Nara, Japan
Email: {thonglek.kundjanasith.ti7, ichikawa}@is.naist.jp, iida@itc.naist.jp
† Tohoku University, Sendai, Japan
Email: keichi@tohoku.ac.jp
‡ Kanazawa University, Ishikawa, Japan
Email: chawanat@staff.kanazawa-u.ac.jp
§ Kasetsart University, Bangkok, Thailand
Email: pattara.l@ku.ac.th

***Abstract*—Federated learning trains a model on a centralized server using datasets distributed over a massive amount of edge devices. Since federated learning does not send local data from edge devices to the server, it preserves data privacy. It transfers the local models from edge devices instead of the local data. However, communication costs are frequently a problem in federated learning. This paper proposes a novel method to reduce the required communication cost for federated learning by transferring only top updated parameters in neural network models. The proposed method allows adjusting the criteria of updated parameters to trade-off the reduction of communication costs and the loss of model accuracy. We evaluated the proposed method using diverse models and datasets and found that it can achieve comparable performance to transfer original models for federated learning. As a result, the proposed method has achieved a reduction of the required communication costs around 90% when compared to the conventional method for VGG16. Furthermore, we found out that the proposed method is able to reduce the communication cost of a large model more than of a small model due to the different threshold of updated parameters in each model architecture.**

***Index Terms*—Sparse Communication, Edge Computing, Federated Learning, Neural Networks**

I. Introduction

Currently, there has been a rise in the use of edge devices since edge devices have more computing power than ever [1]. Massive complex structured and unstructured data are being generated and accumulated over a large number of edge devices [2]. The collected data on mobile devices is advantageous for deep learning, which requires a significant amount of data [3]. However, transferring raw data directly from clients to a server is not a preferred procedure as it lacks data privacy and consumes a lot of network resources [4]. Thus, federated learning has been introduced to address these issues. Federated learning is a machine learning method to train a single global model on a centralized server by indirectly using training data distributed across a large number of edge devices. In federated learning, the clients transfer only the trained local models instead of the raw data, and it mitigates privacy and network communication issues [5]. It also allows each edge device to train a client-specific local model, while training the global model [6].

Nevertheless, there is still room for improvement in reducing the communication cost of federated learning, since just transferring the trained models still consumes a significant amount of network resources [7]. Federated learning is executed over a large number of edge devices that are often connected to wireless networks such as cellular networks. Since wireless networks provide lower bandwidth compared to wired networks, communicating the models between the server and clients takes longer and thus limits the frequency of model updates [8]. Thus, by improving the communication efficiency, the global model on the server can be continuously and efficiently updated, and the performance of federated learning can be improved [9].

There are two popular approaches to reduce the required communication cost for federated learning: (1) reducing the number of communication rounds [10], and (2) reducing the amount of communication in each round [11]. However, if the number of communication rounds is reduced, the global model may miss local training information that could have been obtained in the omitted rounds, and thus the model cannot be continuously updated [12]. Reducing the communication cost in each round can also be divided into two approaches: (1) transferring whole models from selected clients [13] and (2) transferring compressed models from all clients [14]. In particular, transferring compressed models is a promising approach because it can balance the trade-off between communication cost and model accuracy by adjusting the compression ratio [15].

In this paper, we propose a method to transfer sparse models instead of dense models for reducing the communication cost on both uplink and downlink in federated learning. The proposed method constructs a sparse model by selecting only parameters that have been updated significantly. We compute the absolute difference between the parameters of the local model before and after training, and exchange only the upper quantile of the updated parameters between the server and the clients. This parameter-wise selection approach increases the opportunity to reduce the communication cost since it omits the unnecessary parameters and keeps the necessary parameters for transfer. It is reasonable not to transfer the parameters

Algorithm 1 Federated averaging (ServerExecutes)

Parameter: R is the number of communication rounds, N is the number of clients, C is the fraction of selected clients, w_r^n is the local model for client n at round r, w_r is the global model at round r

```
1: Initialize $w_0$
2: for $r = 1, 2, \ldots, R$ do
3:     $S_r \leftarrow$ A random set of $C \times N$ clients
4:     for $n \in S_r$ do
5:         $w_{r+1}^n \leftarrow$ ClientsUpdate($w_r$)
6:     end for
7:     $w_{r+1} \leftarrow \frac{1}{N} \sum_{i=1}^{N} w_{r+1}^i$
8: end for
```

Algorithm 2 Federated averaging (ClientsUpdate)

Parameter: E is the number of epochs, B is the number of batch sizes, w_r^n is the local model for client n at round r

```
1: Receive $w_r$ from the server
2: Initialize $w_{r+1}^n \leftarrow w_r$
3: for $e = 1, 2, \ldots, E$ do
4:     for $b = 1, 2, \ldots, B$ do
5:         $w_{r+1}^n \leftarrow w_{r+1}^n - \eta g_n$
6:     end for
7: end for
8: Transfer $w_{r+1}^n$ to server
```

that do not have significant updates in the local model, since they may not have much impact on the global model update. Additionally, the proposed method allows adjusting the trade-off between the model accuracy and the communication cost with a hyperparameter. This hyperparameter controls the level of sparsification, *i.e.* what fraction of the model are exchanged between the server and clients.

The remainder of this paper is structured as follows. Section II reviews related literature on reducing the required communication cost for federated learning. Section III describes the proposed method to transfer sparse models instead of dense models for federated learning. Section IV shows the experimental results when applying the proposed method to various neural network models with different datasets. Lastly, Section V concludes this paper and discusses our future work.

II. Background

This section gives a brief overview of existing techniques to reduce the communication cost in federated learning.

A. Federated learning algorithms

Federated Stochastic Gradient Descent (FedSGD) is a popular conventional federated learning algorithm based on SGD [16]. The state-of-the-art of federated learning algorithm is Federated Averaging (*FedAVG*) [17]. FedAVG was developed to reduce the communication costs of FedSGD and improve the convergence speed [18]. The number of local training epochs performed on the client is increased compared to FedSGD.

Algorithm 1 and 2 show the pseudocodes of FedAVG for the server and the client, respectively. On the server side, a subset of participating clients is randomly selected in each round. Next, the server receives the local model w_{r+1}^n from each selected client n. The server then computes the new global model w_{r+1} by averaging the local models received from the clients. On the client side, the client receives the latest global model w_r from the server. The client updates the local model w_{r+1}^n using its local training data. Here, η and g denote the learning rate and the gradient, respectively. Each client sends the updated model w_{r+1}^n to the server. FedSGD can be considered as a special case of FedAVG where the local batch size is ∞ and the number of local training epochs is one. In addition, FedAVG allows assigning the fraction of selected clients to reduce the number of participated clients in each communication round.

B. Reducing communication costs in federated learning

We categorize the existing methods for reducing communication costs in federated learning into two approaches: (1) transferring uncompressed models from selected clients and (2) transferring compressed models from all clients.

The most common approach for transferring the whole model from selected clients is constructing the criteria to decide which clients should be selected. Wu et al. proposed a method to measure the similarity between the global model and the local model on each client using the gradient distribution [19]. They scored each client and adjust its participation probability of each client using the similarity score between the local and global models. Park et al. proposed *FedPSO* [20] to reduce the number of selected clients in each round using particle swarm optimization. They used particle swarm optimization to select the clients based on their loss values of local training. However, the loss value is not a good performance indicator for representing the performance of a local model since the local training datasets are often not identically distributed across the clients.

Yao et al. proposed *FedMMD* [21] to select the clients that participate in each communication round using maximum mean discrepancy, a distance measure in the probability space. They computed the maximum mean discrepancy of a local model before and after the local training, and transfer the local model to the server only if the maximum mean discrepancy reaches a threshold.

Compared to the method of deciding whether to select or not on a client-by-client basis, the model compression methods can select data to be transferred at a finer granularity. We, therefore, focus more on the model compression methods for federated learning in this paper. Konnecy et al. worked on compression of transferred models using techniques such as *Low Rank Approximation*, *Random Masking*, and *Quantization* [15]. However, they compressed uplink communication only since the uplink is typically slower than the downlink.

Algorithm 3 Proposed method (ServerExecutes)

Parameter: R is the number of communication rounds, N is the number of clients, w_r^n is the local model for client n at round r, w_r is the global model at round r

```
1: Initialize $w_0$
2: for $r = 1, 2, \ldots, R$ do
3:     for $n = 1, 2, \ldots, N$ do
4:         $w_{r+1}^n \leftarrow$ ClientsUpdate($w_r^n$)
5:     end for
6:     $w_{r+1} \leftarrow \frac{1}{N} \sum_{i=1}^{N} w_{r+1}^i$
7:     for $n = 1, 2, \ldots, N$ do
8:         for $p \in$ parameters of $w_{r+1}^n$ do
9:             if $p$ is not NULL then
10:                $w_{r+1}^n[p] \leftarrow w_{r+1}[p]$
11:            else
12:                $w_{r+1}^n[p] \leftarrow$ NULL
13:            end if
14:        end for
15:    end for
16: end for
```

In contrast, our method considers reducing communication cost in each round by transferring the compressed model from all clients, and also reducing the communication cost of both uplink and downlink.

Aji et al. introduced sparsification of models for distributed learning [22]. Their proposed method exchanges only the top updated parameters of the global and local models. The server then transfers the same sparse global model to all clients. However, we propose a method to exchange different sparse models for each client in order to improve the performance of federated learning in terms of both communication efficiency and accuracy. The main difference between distributed learning and federated learning is that distributed learning proposes to achieve high accuracy over a single dataset, while federated learning proposes to achieve high accuracy over multiple datasets across the clients. Therefore, our proposed method transfers different sparse models in order to optimize the models to the local dataset on each client.

III. Methodology

The basic idea behind the proposed method is to sparsify the models exchanged between the server and clients. Parameters in neural networks are usually constructed as a graph data structure where each node does not update its weight as others [23]. The most updated parameters might have the most impact on model performance since it affects the cumulative changing of the graph more than the less updated parameters. Thus, we compute the absolute difference of model parameters before and after local training, and construct a sparse model that contains only the upper quantile of updated parameters. The server constructs a sparse global model for each client with the same sparsification pattern as the local model received from the client, and sends it to the client.

Algorithm 4 Proposed method (ClientsUpdate)

Parameter: E is the number of epochs, B is the number of batch sizes, Q is the quantile, w_r^n is the local model for client n at round r

```
1: Receive $w_r^n$ from the server
2: for $p \in$ parameters of $w_r^n$ do
3:     if $p$ is NULL then
4:         $w_r^n[p] \leftarrow w_{r-1}^n[p]$
5:     end if
6: end for
7: Initialize $w_{r+1}^n \leftarrow w_r^n$
8: for $e = 1, 2, \ldots, E$ do
9:     for $b = 1, 2, \ldots, B$ do
10:        $w_{r+1}^n \leftarrow w_{r+1}^n - \eta g_n$
11:    end for
12: end for
13: threshold $\leftarrow$ Quantile($|w_r^n - w_{r+1}^n|, Q$)
14: for $p \in$ parameters of $w_{r+1}^n$ do
15:     if $|w_r^n[p] - w_{r+1}^n[p]| <$ threshold then
16:         $w_{r+1}^n[p] \leftarrow$ NULL
17:     end if
18: end for
19: Send $w_{r+1}^n$ to the server
```

Algorithm 3 shows the pseudocode of the proposed method on the server. At the initialization step, the server distributes the global model to all clients. The server then receives the sparse local model from each client (line 4). The sparse local models are aggregated to a dense global model using averaging (line 6). Next, a sparse global model is created by replacing the parameters of the sparse local model received from the client with the updated parameters of the global model (line 10). The same sparsification pattern is maintained by leaving the dropped parameters (p is NULL) in the local model as NULL (line 12). The sparse local model is compressed using gzip[1].

Algorithm 4 shows the pseudocode of the proposed method on the clients. Each client receives the sparse global model from the server (line 1). Next, a dense local model is constructed by replacing the NULLs in the received sparse global model with the dense local model from the previous round (line 2–7). The dense local model is then trained using the local dataset (line 8–12). We then compute the threshold for sparsification. We compute the absolute differences of parameters before and after local training, and use their Q-quantile value as the threshold (line 13). The parameter Q is supplied by the user to adjust the communication cost and model accuracy. For instance, if Q is set to 0.9, the top 10% of the parameters are selected for transfer. Afterwards, a sparse local model is constructed by replacing the parameters less than the threshold with NULLs (line 14–18). The sparse local

[1]https://www.gnu.org/software/gzip/

TABLE I: Experimental Setup

Configuration	Value
# of communication rounds (R)	10
# of clients (N)	10
# of local epochs (E)	5
# of local batch sizes (B)	8

TABLE II: Model specification

Name	Size [MB]	# of Parameters
VGG16	553.43	138,357,544
ResNet152	243.21	60,419,944
DenseNet201	82.92	20,242,984
MobileNet	17.02	4,253,864

model is then compressed using gzip. At last, the compressed sparse local model is sent to the server (line 19).

In sparse distributed learning, the same global model is broadcasted to all clients. In constrast, our proposed method for federated learning sends the global model to each client with different sparsification patterns. In our method, the sparsification pattern of the global model depends on the sparsification pattern of the local model received in the previous communication round. It means that only the parameters sent to the server due to large updates confirmed in the local training will be updated back with the parameters of the aggregated global model. This prevents unnecessary updates of the local model and also helps the local model maintain high accuracy on its local dataset.

IV. Evaluation

This section evaluates our proposed method from four aspects. First, we compare the proposed method to the sparse communication method for distributed learning. Second, we compare the proposed method to existing communication reduction methods for federated learning. Third, we evaluate the proposed method using four state-of-the-art neural network models for image classification task to assess the impact of the model size on the performance of the proposed method. Lastly, we investigate if the proposed method can be applied to different datasets.

A. Experimental environment

We measured the communication cost and accuracy of the global model to evaluate the practical applicability of the proposed method. The communication cost on both uplink and downlink communication was measured by estimating the number of bytes transferred over the network connection. All sparse global and local models were stored for each communication round. Then, for each communication round, we calculated the size of each sparse local model and the size of sparse global model multiplied by the number of participating clients to estimate the total number of transferred bytes. The experimental setup for federated learning is shown in Table I. The experimental environment was set up using Docker, and the server and clients were executed in separate containers.

We conducted our experiments using four neural network models and four datasets. Table II shows the specifications of the models used to evaluate the proposed method, VGG16, ResNet152, DenseNet201, and MobileNet. These four models represent state-of-the-art image classification models of different scales. Although large-scale models can achieve higher accuracy than small-scale models, not all edge devices can deploy large scale models due to resource constraints. Thus, we evaluated the proposed method using models with different scales.

The experiments were conducted using CIFAR10, CIFAR100, MNIST, and FMNIST datasets. These four datasets are the most popular datasets used to evaluate image classification tasks. CIFAR10 and CIFAR100 consist of 60,000 images (50,000 training samples and 10,000 testing samples) each of which is a 32×32 pixel 3-channel image. MNIST and FMNIST consist of 70,000 images (60,000 training samples and 10,000 testing samples) each of which is a 28×28 pixel single-channel image. The training samples are distributed equally to each client, and the testing samples are stored on the server to evaluate the accuracy of the global model.

B. Comparison to distributed learning

We compared the performance of the proposed sparse communication method for federated learning and that of the existing method for distributed learning [22]. Here, distributed learning refers to the setup where the server sends the same sparsified model to all clients, whereas in the proposed method the server sends a different sparsified model to each client. While distributed learning focuses on training a global model

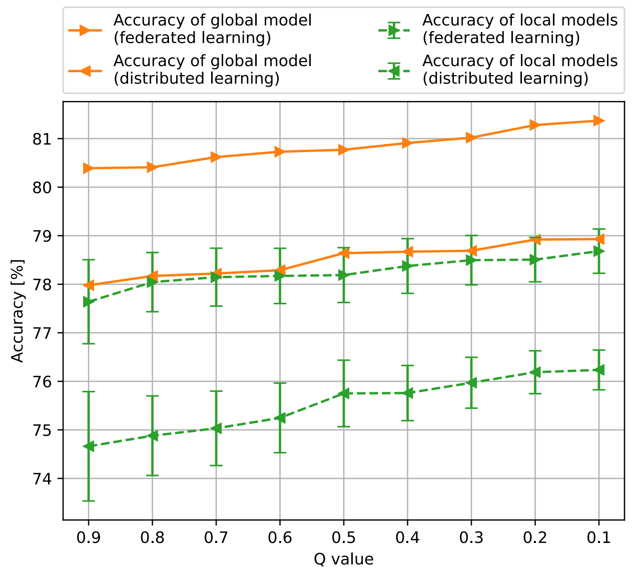
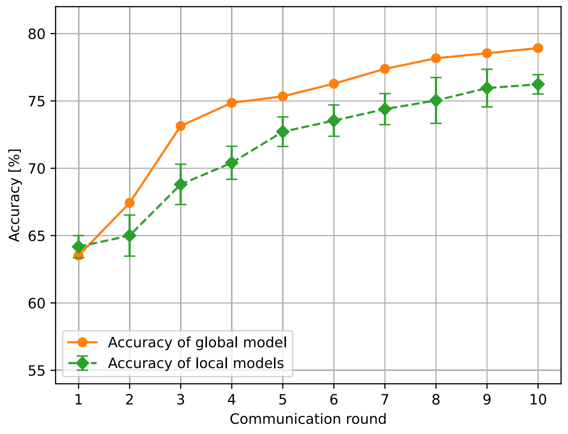
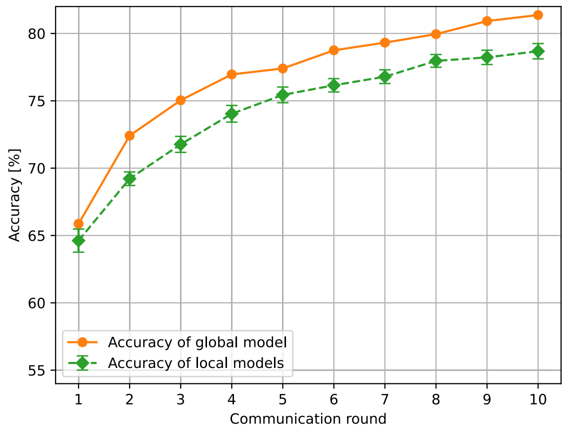


Fig. 1: Accuracy of global and average accuracy of local models between federated and distributed learning



(a) Distributed learning



(b) Federated learning

Fig. 2: Comparison of sparse communication methods for federated and distributed learning ($Q = 0.1$)

to achieve high accuracy over a single dataset, federated learning is expected to maintain the accuracy of both global and local models over the global and local datasets. Thus, we measured the accuracy of global model and the average accuracy of local models between our federated learning and the existing distributed learning. The global model was evaluated using the testing samples on the server and the local model was evaluated using the training samples on its own local device.

Figure 1 presents the accuracy of global model and average accuracy of local models between federated learning and distributed learning in the case of VGG16 model on CIFAR10 dataset while varying the value of the hyperparameter Q, which adjusts the communication cost. We found out that both the accuracy of the global model the average accuracy of local models of our proposed method are higher than that of the existing distributed learning. Moreover, we found out that the variance of local model accuracy of the distributed learning is larger than that of our method as shown in Fig. 1.

Figure 2 shows the accuracy of global and local models at each communication round for the existing distributed learning and our federated learning. In the distributed learning, we observed a large variance in the accuracy of the local model during the training process as shown in Fig. 2(a). The global model with the same sparsification pattern used in distributed learning reduces the accuracy of the local model because the local model is initialized with the generalized global model for each communication round. On the other hand, the accuracy of local models of our federated learning is more stable than that of distributed learning as shown in Fig. 2(b). In the federated learning, the accuracy of global model is improved continuously unlike in the distributed learning.

Although distributed learning, which aims to achieve high accuracy with only a single global dataset, is expected to achieve higher accuracy in the global model, the results show that our federated learning model achieved higher accuracy in the global model as well. This may be due to the fact that distributed learning, which transfers the same global model to all clients, is not able to build highly accurate local models, which also leads to a decrease in the accuracy of the global model.

C. Comparison to the existing methods

We compared the performance of the proposed method and existing learning methods for communication reduction: FedAVG, FedPSO, FedMMD, Low rank approximation, Random masking, and Quantization. Figure 3 shows the relative communication cost and global model accuracy for VGG16 model on CIFAR10 dataset. Each method can be adjusted for accuracy and communication reduction by hyperparameters. Table III shows the hyperparameters of each method. The original communication cost indicates the total communication cost when FedAVG is applied with C set to 1.0.

FedPSO, FedMMD, Low rank, Random mask, and Quantization do not reduce communication cost by more than 50% since they reduce the uplink communication only. FedAVG and the proposed method are the only two methods that can reduce the communication cost by more than 50%. However, the accuracy of the global model in FedAVG decreases sharply when reducing the fraction of the selected clients. On the other hand, the accuracy of the global model in the proposed method decreases slowly when increasing Q.

The results show that the proposed method outperforms FedAVG in terms of the required communication cost and the accuracy of the global model. FedAVG decreases the number of selected clients to reduce communication cost, and thus some local datasets are not used in the training. On the other hand, the proposed method does not eliminate the number of clients, but instead replaces the transfer of dense local models with sparse local models, and removes parameters that are less updated in the model.

D. Results for different models

We investigated the impact of model size on the reduction of the required communication cost and global model accuracy using the proposed method and the existing methods. The

TABLE III: Learning methods and hyperparameters

Name	Hyper-parameter	Description
FedAVG	C	The fraction of clients selected in each communication round. All clients will participate in the communication round if C is set to one.
FedPSO	α	The inertia weight of swarm optimization for each client during optimization. Varying α value does not reduce the communication cost. We set α to 0.1 which produce the highest accuracy.
FedMMD	λ	The coefficient of MMD loss between the global and local models. The clients have less MMD loss compare than the previous round will be selected.
Low rank approximation	K	The rank of the low-rank matrix to be converted.
Random mask	M	The size of random mask to generate a random sparsification pattern.
Quantization	B	The quantized bit used for bit-quantization.

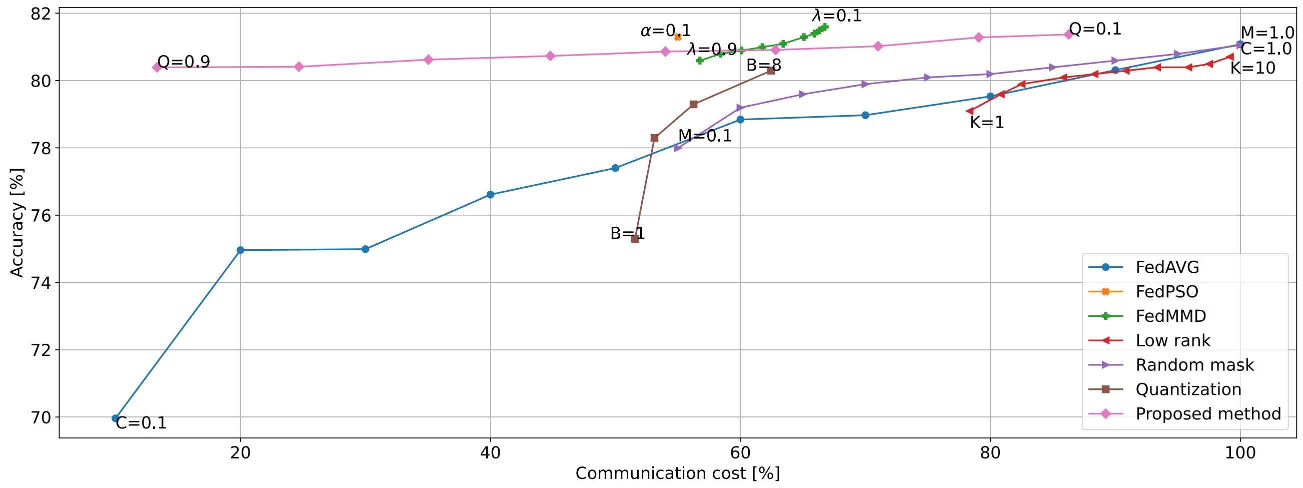


Fig. 3: Accuracy of global model and total communication cost (VGG16 model and CIFAR10 dataset)

experiments were conducted for the model with different size on the CIFAR10 dataset.

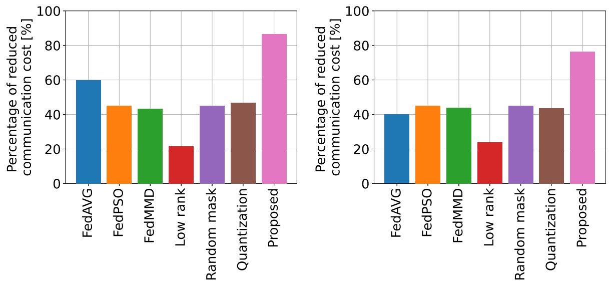
Figure 4 presents the optimal results of the communication cost reduction ratio of the proposed method and the existing methods for four image classification models (VGG16, ResNet152, DenseNet201, and MobileNet) when the accuracy of the global model satisfies an acceptable accuracy. Here, the acceptable accuracy is defined as 5% lower than the original accuracy. The results indicate that the reduction of the communication cost from FedPSO, FedMMD, Low rank approximation, Random mask, and Quantization are almost identical for all model architectures. Low rank approximation achieves the least reduction when compared to other methods.

FedPSO and FedMMD reduce the communication costs by building criteria to reduce the number of clients selected to only those that fulfil the criteria. The number of selected clients based on the criteria will be a small number as one to three clients in each communication round. This number of selected clients does not vary much in FedPSO and FedMMD. Therefore, the communication cost of those methods does not vary much either since the variation of the communication cost depends on the number of selected clients. On the other hand, Low rank, Random Mask, and Quantization are compression techniques that do not consider the model architecture, and the compression ratio depends on the number of redundant parameters, not the model architecture. Thus, the reduction in communication costs through these methods is similar for all models. However, the communication cost reduction by FedAVG and the proposed method depends on the model architecture. The communication cost reduction by the proposed method outperforms the others in VGG16 and ResNet152, but not so much in DenseNet201 and MobileNet.

E. Results for different datasets

In this evaluation, we measured the average of transferred model size against the accuracy of the global model by varying the dataset since we aim to investigate the performance of the proposed method to reduce communication cost on different model architectures and datasets. The average of transferred model size represents how much the proposed method is able to reduce the size of each architecture model for transferring between a server and the clients.

Figure 5 shows the global model accuracy and average transferred model size for different model architectures and datasets. Each line connects 18 markers representing different Q values. In this evaluation, we varied Q from 0.1 to 0.9 at an



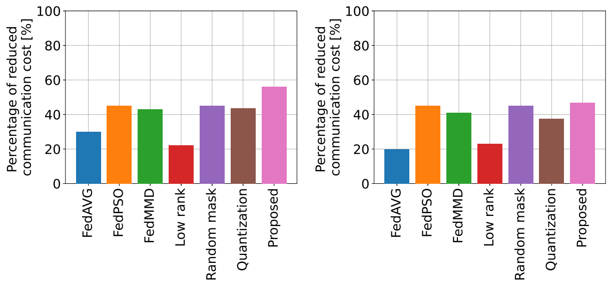
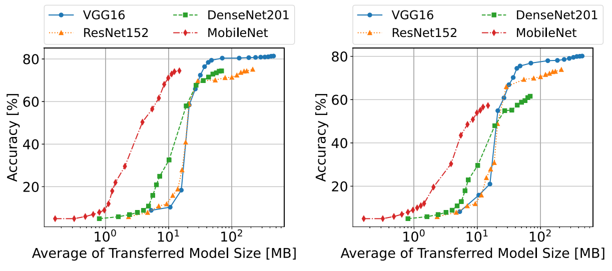


Fig. 4: Percentage of reduced communication cost for different models



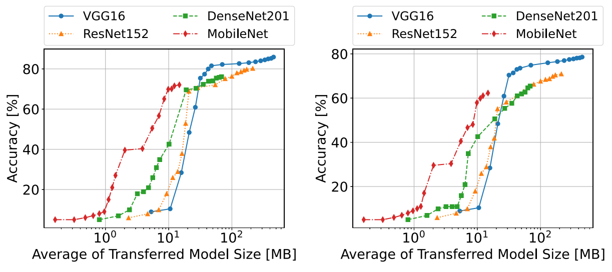


Fig. 5: Average transferred model size for each model architecture with the different datasets

interval of 0.1, and 0.91 to 0.99 at an interval of 0.01. In the case of CIFAR10, the proposed method reduced the average transferred model size of VGG16 from 553 MB to 32 MB (94.21% of reduction), while the global model accuracy was 76.39%. This is slightly (1.93%) higher than the global model accuracy of MobileNet when the average transferred model

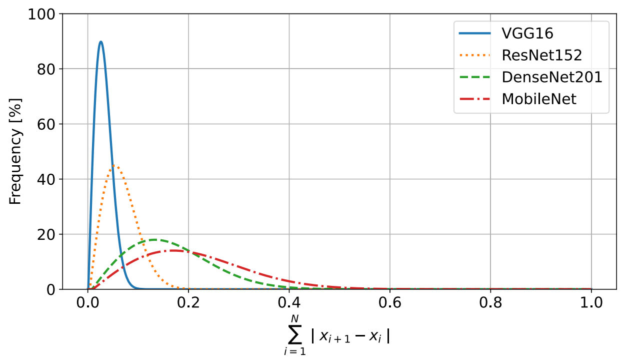


Fig. 6: Distribution of parameter updates for each model

size is 17 MB as shown in Fig. 5(a). If the proposed method reduces the average transferred model size of VGG16 with the same reduction size in the case of CIFAR100 dataset then the global model accuracy of VGG16 is significantly higher than the global model accuracy of MobileNet by 9.64% as shown in Fig. 5(b). In addition, we observed that the global model accuracy of VGG16 is slightly higher than the global model accuracy of MobileNet by 3.32% in the case of MNIST dataset but the global model accuracy of VGG16 is significantly higher than the global model accuracy of MobileNet by 9.10% as shown in Fig. 5(c) and Fig. 5(d).

Application requirements and resource limitations need to be considered when selecting the model to deploy on the clients and server. The selected model must meet the requirements in terms of computing, storage and network resources of every edge device, and achieve the model accuracy required by the application. Typically, a larger model is able to achieve accuracy higher than a smaller model. The proposed method can reduce the communication cost of a large model until consume the network resources slightly larger than a small model while maintaining significantly higher accuracy.

F. Distribution of parameter updates

The proposed method reduces the communication cost of a large model more than a small model with most models. Figure 6 plots the frequency of updated values per communication round and per client for each model architecture. We found out that the frequency of small updated parameters in a larger model is higher than in a smaller model. Updating parameters in neural network models represents how much the model changes, which means small models require more updating than large models [24]. Therefore, the proposed method can reduce the communication cost of large models more than small models since large models update the parameters with the lower threshold of parameters than small models.

Generally, a larger model achieves higher accuracy than a smaller model [25]. However, the proposed method reduces the communication cost of a large model to be slightly larger than of a small model while a smaller model has slightly lower accuracy on CIFAR10 and MNIST datasets. Furthermore, we found out that the proposed method is able to reduce the communication cost of a large model to be slightly larger than of a small model while a smaller model has significantly lower accuracy on CIFAR100 and FMNIST datasets. Hence, deploying the model on the edge devices for federated learning depends on the use case scenario or resource constraints.

V. Conclusion

In this paper, we proposed a novel method to reduce the communication cost for federated learning on both uplink and downlink communication. The proposed method utilizes exchanging the most updated parameters of neural network models. We used diverse models and datasets to evaluate the proposed method in terms of model accuracy and communication cost. The proposed method achieved a reduction in the communication costs approximately 90% compared to the traditional method for VGG16. Moreover, we found out that the proposed method reduces the communication cost of larger models more than smaller models since the threshold of updated parameters in a large model is greater than in a small model. In some cases, a small model achieve slightly lower accuracy than a larger model with a slight difference in communication cost.

A future work is to investigate other neural network models to improve reducing the required communication cost for federated learning while maintaining the accuracy of a global model. Updating the parameters in other neural network models should be observed during the local training procedure on each local edge devices for reducing communication cost efficiently. In addition, larger number of edge devices should be used to evaluate the performance of the proposed method.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Numbers JP18K11326 and JP20K19808.

References

[1] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 110–115, Nov 2018.

[2] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, Apr 2015.

[3] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digital Communications and Networks*, Jun 2021.

[4] Z. Du, C. Wu, T. Yoshinaga, K. Yau, Y. Ji, and J. Li, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Open Journal of the Computer Society*, vol. 1, no. 01, pp. 45–61, Jan 2020.

[5] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, Sep 2020.

[6] 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.

[7] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, Oct 2019, pp. 954–964.

[8] D. Conway-Jones, T. Tuor, S. Wang, and K. K. Leung, "Demonstration of federated learning in a resource-constrained networked environment," in *Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP)*, Aug 2019, pp. 484–486.

[9] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive internet of things for smart industries," *IEEE Access*, vol. 8, pp. 168 854–168 864, Sep 2020.

[10] J. Hamer, M. Mohri, and A. T. Suresh, "FedBoost: A communication-efficient algorithm for federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 3973–3983.

[11] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, "Federated learning with additional mechanisms on clients to reduce communication costs," *CoRR*, vol. abs/1908.05891, Aug 2019.

[12] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 869–904, Jan 2020.

[13] J. Mills, J. Hu, and G. Min, "Communication-Efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, Jul 2020.

[14] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proceedings of the NIPS Private Multi-Party Machine Learning Workshop*, Oct 2016.

[15] S. K. Lo, Q. Lu, C. Wang, H.-Y. Paik, and L. Zhu, "A systematic literature review on federated machine learning: From a software engineering perspective," *ACM Computing Surveys*, vol. 54, no. 5, Jun 2021.

[16] J. Rie and Z. Tong, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, vol. 26, Dec 2013, pp. 315–323.

[17] Y. Li, T.-H. Chang, and C. Y. Chi, "Secure federated averaging algorithm with differential privacy," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Oct 2020, pp. 1–6.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIStat)*, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[19] W. Wu, L. He, W. Lin, R. Mao, C. Huang, and W. Song, "FedProf: Optimizing federated learning with dynamic data profiling," *CoRR*, vol. abs/2102.01733, Feb 2021.

[20] S. Park, Y. Suh, and J. Lee, "FedPSO: Federated learning using particle swarm optimization to reduce communication costs," *MDPI Sensors Journal*, vol. 21, no. 2, Jan 2021.

[21] X. Yao, T. Huang, C. Wu, R.-X. Zhang, and L. Sun, "Towards faster and better federated learning: A feature fusion approach," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Aug 2019, pp. 175–179.

[22] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sep 2017, pp. 440–446.

[23] C. Zhang and W. Xu, "Neural networks: Efficient implementations and applications," in *Proceedings of the IEEE International Conference on ASIC (ASICON)*, Oct 2017, pp. 1029–1032.

[24] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[25] A. Brutzkus and A. Globerson, "Why do larger models generalize better? A theoretical perspective via the XOR problem," in *Proceedings of the International Conference on Machine Learning (ICML)*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, 09–15 Jun 2019, pp. 822–830.