Masked Transmitting for Federated Learning

Da Wei
AI Institute, Tongdun Technology
da.wei@tongdun.net

Dan Meng
AI Institute, Tongdun Technology
dan.meng@tongdun.net

Hongyu, Li
AI Institute, Tongdun Technology
hongyu.li@tongdun.net

Xiaolin Li
AI Institute, Tongdun Technology
xiaolin.li@tongdun.net

Abstract

Federated learning is a growing privacy framework whose training data is separated into several participants with security care[6]. The separated raw data will be well supervised and its transmission is strictly prohibited by its owner. Due to the information interaction between participants which is required by building a generalized model no matter what kind of situation is during the training process, people proposed the federated learning to exchange information in which extracted from raw data. Because of the information interaction can not be omitted, it becomes the bottleneck of efficiency of the whole system. Bigger the model, larger the exchanged data. We proposed FedMT, a novel, simple, and efficient way to aggregate models from different participants with masked transmission and build a generalized model for federated learning in fewer data exchange. First, every participant will train locally with respective data. Then, they make ranks by calculating filters' contribution values in network layers and construct a mask based on the rank. The unmasked gradients will be sent to a server and making aggregation with unmasked gradients from other participants. Finally, the aggregated gradient will be sent back to all the participants and update the model respectively. Our result shows that the proposed method FedMT reduce a great amount of communication data while maintaining competitive performance compared with baseline. Our method aggregates the high contribution value from the other participants which not only enhances the global generalize ability but also maintains the local fit ability.

1 Introduction

In recent years, with the thriving of smart devices and sensors, the method of data collection is rapidly growing. Researchers start to study people's behavior based on the collected data. At the same time, some privacy policy like GDPR(General Data Protection Regulation)[1] has been issued by governments for protecting personal privacy. A paradox shows up between getting more data to train a better model and restricting data exchange to secure data privacy. As a special case of distributed machine learning approaches, Federated Learning (FL) enables end devices to collaboratively learn a shared model while keeping all the training data on the device. Like the method of FL first mentioned by Google for edge calculation, FL for companies plays similar work like edge calculation. Every participant is a client in FL architecture and the server is a party to aggregate gradients from clients. This computing paradigm provides a privacy-preserving mechanism to leverage massive decentralized computing resources and user data to power more intelligent applications.

The original FL framework works simply. First, FL trains local models based on local data. Secondly, transmitting gradients of models to server for aggregating gradients from all the clients. Thirdly,

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

after aggregation, the server sends aggregated gradients back to every client to the next training step. Repeating previous steps until model coverage in every client. There are several shortcomings in this simple and raw workflow. People may re-engineer the global neural network with the total gradients to get privacy data in clients; the non-i.i.d distribution of data in different participants leads the generalized model unconverged; heavy burden for the communication system with large network and so on. After ciphering the gradients and solving the re-engineering problem, the encrypted data need times of bandwidth for exchanging data. Data exchange consumption becomes way higher than the calculation consumption. For solving the problem of bandwidth overload and non-i.i.d data distribution, we proposed the FedMT which can upgrade the model with the outstanding data from different clients and reduce the amount of transmitting data by masking less contribution filters in the neural network.

Following statement will show the requirement of bandwidth resources with calculation. The number of clients in FL systems may in millions and the local model will get million Bytes. For example, VGG, the widely known neural network for image recognition has 160M parameters, weighting 25MB when represented by 32 bits. 10^{12}Bytes exchanged data for one gradient aggregation without any ciphering method. Even if the encrypted gradients need ten times of space, the exchange data will be Exponential. It is an unaffordable burden for the network of central server. The proposed method will calculate the contribution value of all the filters in the neural network to getting the global contribution. By the way, the number of transmission filters can be controlled manually in which be able to adjust for specific situations.

Stop sending the redundancy gradient by masking the low rank filters not only decreasing the bandwidth recourse requirement but also protecting the less important filters which is fitted at local model. If we simply aggregate the well fitted filters with FedAvg like simple FL do, the model will always be a under-fitting circumstance. This is how the non-i.i.d situation hurt the converge processing while training. For protecting the well fitting and upgrade the under-fitting filters, we should aggregate the filters while they need upgrade and reduce the affecting of high gradients from other clients.

HCA(High Contribution filter Aggregation) is our aggregation method based on the transmitted data in FedMT. It only aggregates the gradients for clients unmasked filters at the central server. Once the filter is masked by local, the aggregation step at the server will not add it as a contributor to the upgrading sequence. When the updated gradients comeback to local, local devices will also run the aggregation step which will be mentioned at Section3.2. The separated aggregation method will maintain the local filter stably, meanwhile, it can subscribe to the high contribution gradients to strength generalization capabilities. This workflow collect gradient with aggregation instead of abandoning them as most pruning paper does. This accumulates the small contribution until the model needs it and update filters which is important in other clients.

Based on summarizing previous work, we propose a novel, simple and efficient method to relieve FL communication pressure and fixing non-i.i.d problem, namely FedMT. Our starting point is that parameters with larger gradients have larger contributions to update the model, thus the priority of transmission is also higher. In each iteration, the gradients of all parameters are obtained locally and sorted in descending order. We record the gradient of the previous part and the corresponding index and send it to the server. Experiments have proved that while FedMT significantly relieves communication pressure. Comparing with FedAvg(average aggregation at the central server), the accuracy of FedMT is competitive.

Our contribution is to propose a masked gradient transmission mechanism which reduces transmit burden significantly. At the same time, the new aggregation method HCA solves the non-i.i.d situation between participants.

2 Related Work

2.1 Federated Learning

Federated learning is a machine learning framework that can effectively help multiple institutions to perform model training with privacy data under the requirements of user privacy protection, data security, and government regulations.[9] For different situation, federated learning is divided into

cross-sample federated learning (horizontal federated learning), cross-feature federated learning, and hybrid federated learning (federated transfer learning)

In cross-sample federated learning, when the overlap of the user feature is more than the overlap of user sample in ratio, we re-build the dataset according to the cross-sample diagram which extracts the two users with the same characteristics but not the same users and train.

In cross-feature federated learning, when the overlap of the user sample is more than the overlap of user feature in ratio, we re-build the dataset according to the cross-feature diagram which extracts the two users with the same samples but not the same features and train.

In hybrid federated learning, when the user samples and user features of participants' datasets are less overlapped, we will not do any preprocessing on data, but use transfer learning to overcome the shortage of data or labels.

2.2 Communication Burden Reducing Strategies

There are several relevant approaches to cut off in different fileds. Working in [8] shows the the constraints of bandwidth which limited the maximum bits remote devices allowed to receive. The physical rules restrict astronomical data transportation. In machine learning implementations, like distributed learning, Federated Learning and so on. Techniques with less communication pressure always have penalties on compute consumption[7].

Approximation, sparsification, sub-sampling, and quantization which converting original data to another formatting can maintain a high performance in training. The mid-product would be simpler and easy to compress. The simple mid-product may leak the information on the server-side.

The final candidate is to release the transmission pressure by sacrificing model complexity. For example, people prune model to get a less complexity model with a smaller size for balance communication and training conflict. Setting 64-bits float to 32-bits to save half bandwidth for efficiency. Ciphering data within 512-bits instead of 1024-bits to raise concurrency.

2.3 Pruning

Pruning is a common and popular technique which is aim to develop smaller, more efficiency and capable neural network in the industrial side. People would like to cut off the leaves of complexity models. Implementing pruning, training process could be faster, less consumption, and devices friendly. Pruned models are the best choice for edge devices which have not powerful computation ability and consumption economic preference.

There are several approaches to pruning the network. The work in [5] focused on cutting off the filters on the CNN layer. It is a great idea to prune the networks' filter which is However, pruning at the filter lever is coarse and wild. The network will lose some outstanding channels which is a simple filter. So, people also working on the method which cut the specific weight off in the model.[3] All the previous work needs extra computation for finding the position of pruning, people pay ones and get times from model pruning.

Instead of focusing on model slimming, pruning also pays attention to the data security problem. For security issues caused by the adversarial attack in the pruned model. The work in [2] shows a trusted approach (ATMC) which integrates three compression mainstreams and solve the challenging constrained problem.

3 FedMT

We build a novel Federated Learning workflow called FedMT to solve the release bandwidth burden and solve the non-i.i.d problem. Appendix A is the global diagram of the whole workflow. The central server initializes the original network and allocates it to all the participants. All the clients train the model with their data as simple training work until the local devices reach the step of doing mask in clients. While doing the mask calculation, the model will get the most contribution filters in the network. Masked the k worst filters, the valuable and unmasked gradients will be sent to the server with its index for the first aggregation. Then, when the local network get the gradients from

the server, it will complete the second aggregation. The diagram will repeat until reaching the final epoch.

3.1 Find Gradient Mask

FedMT needs us to make a mask on the network which will stop the exchange of less important filters. What is the importance of each filter and how can we ranking filters becomes a problem. We cite the ideas in model pruning[4]. Instead of choosing filters randomly, we select filters by calculating its contribution values and then we masked those less important filters and exchange the unmasked gradient for less bandwidth resources requirement. We will set a period based on the rule and do the mask calculation for update the unmasked filters to aggregation. The rule can configure manually according to dynamic variation.

As the pseudo-code shows in algorithm1, when we need to do the mask calculation, we will do the process based on the input data. We do one forward propagation and one backpropagation to get the plain output and gradient of each filter. The contribution value is multiplied by the plain outputs and gradients. We do not use the plain output from data getting through the layer as the contribution value. The plain output only shows the current status of this filter which will fluctuate by the vary of filter parameters. Gradient can show the variety of the filter parameters which can not prove the filter if solid enough for proving the contribution. That's why we combine the plain output and the gradient as the contribution values to rank the filters. The chosen contribution filters both contain contribution right away and the potential possibility.

Algorithm 1 Calculate Gradient Masked for the layers

Input: D^n is the input data of the layer, Y is the label of the layer, k is the number of filters to mask.
Parameter: A, G is the activation and the gradient values for all the filters. L is the loss value of the layer.
Output: C is the contribution values for ranking the contribution of each filters in the layer.

1: **while** Do Gradient Mask **do**
2: Make forward propagation through this layer from D^n to A.
3: Getting L from the calculated activation value A and label Y by $L = Y - A$.
4: Do back propagation with loss L and get G.
5: Calculating $C = A * G$ and ascending the contribution values.
6: **end while**
7: **return** C

3.2 HCA (High Contribution filter Aggregation)

Traditionally, the aggregation step will be completed at the central server. Unlike the traditional aggregation method in the Federated Learning framework, our proposed HCA separates the central aggregation into two steps which are illustrated at algorithm2 in the central server and algorithm3 in local.

At first, the unmasked data is sent to the central server for aggregation. Because of the masked transmission mentioned in section 3, the central server can not get all the gradients' information for one specific client. If we apply the average aggregation like FedAvg do in the masked gradients, it is unfair to those filters only get gradients from partial clients. So, our algorithm2 only counting the unmasked gradients as the divisor. The gradient for a particular filter is only related to those clients which unmask the filter. This part is almost like the aggregation method in FedAvg.

It is time to send the gradients back to locals. Masked filters will receive the gradient from other unmasked clients. If we just abort those unmasked gradients, we'd better remove it earlier before it is transmitted to masked clients. It is foolish to get rid of those valuable unmasked gradients, but bandwidth efficiency. We have already take care of bandwidth efficiency by transmitting partial filters instead of all the filters in the neural network. We can config the k value to get less bandwidth requirement.

Algorithm 2 Aggregation for one filter at central server

Input: G_j is the gradient from j-th client, n is the total number of clients which unmasked this filter.

Output: G^s is the aggregated gradients in server.

1: $G^s = \frac{\sum_{j=1}^{n} G_j}{n}$
2: **return** G^s

Now, we want to maintain the performance on accuracy and even get a better one. We keep the unmasked gradients at masked clients. Instead of overlapping original gradients, we build an activation function for aggregating the network.

$$G^l = \frac{Sigmoid(G^s) * G^s + G}{2} \tag{1}$$

Using the $Sigmoid(G^s)$ as the activator to activate itself, the masked clients can get information based on the importance of unmasked gradients. When other unmasked clients are really important, the activation level will be high, this function can get mean gradients from local and server. If unmasked gradient can not activate function well, masked clients can keep their gradients for specific optimization.

Algorithm 3 Aggregation for one filter at local

Input: G^s is the gradient of this filter from central server, G is the gradient before aggregation.

Output: G^l is the aggregated gradients at local.

1: **if** This filter is unmasked by local **then**
2: $\quad G^l = G^s$
3: **else**
4: $\quad G^l = \frac{Sigmoid(G^s) * G^s + G}{2}$
5: **end if**
6: **return** G^l

For the local aggregation in unmasked clients, the final gradients will be good as the gradient from the server.

This two steps aggregation method inherent the idea of the FedAvg. Based on that, we add some non-linear activation on that to solve the partial aggregation problem. Meanwhile, constructing the boundaries in gradients aggregation, the model is easier to underfit while data is at non-i.i.d circumstances.

4 Experiment

We perform some experiments on different networks and datasets for accuracy and make comparison with FedAvg. We run our experiments on Ubuntu16 with 4 Nvidia1080, python3.7, pytorch1.0.1. We stimulate the distributed communication by local communication with python package pydist.

4.1 Training Details

Here are the training parameters we used in the experiment. For variable control, we maintain all the training process with hyper-parameters as following: epoch 100, learning rate 0.01, batch size 64, optimizer SGD, momentum 0.9, weight decay 5e-4. The network is initialized at the central server once per experiment and allocate to all the clients to promise all the local model start at the same position. For every 10 epoch, we will redo the mask calculation process to update the aggregation filters which balance the time consumption of mask calculation and mask upgrade. We only transmit the filters in the CNN layer for exchange which means all the filters in other layers will be abandoned in our experiment. The transmit part we mentioned in table1 stands the transmitted CNN layer which is way smaller than the part of the whole parameters of the network. The experiment of FedAvg aggregate all the parameters the network has.

4.2 Analysis

Transmit Part	AlexNet(Acc)	VGG16(Acc)	Transmit Part	AlexNet(Acc)	VGG16(Acc)
20%	99.300	99.460	20%	67.645	76.555
40%	**99.320**	99.463	40%	**67.748**	76.278
60%	99.297	99.457	60%	67.610	76.508
80%	**99.320**	99.453	80%	67.585	76.878
100%	**99.320**	**99.510**	100%	67.608	76.300
FedAvg	99.187	99.420	FedAvg	66.973	**77.003**

Table 1: Transmit Partial in the table shows that how much percentile of gradient values will be transmitted to the server to upgrade the generalized model. The former table illustrates the test accuracy differential on the MNIST dataset. The highlight number is the best one for each column. The Transmit part is the transmit percentile of CNN layers in mentioned network. The latter one shows the test accuracy on the Cifar10 dataset.

We explore the affection of partial transmission with FedMT, table 1 shows that different percentile of transmission have stable accuracy results even though we just left 20% filters to aggregation. This outcome leads to the conclusion that plenty of redundancy in the network to exchange not only overwhelm the communication system, but also useless for enhancing the model performance. Also, the top accuracy performance of each column almost show at the bottom of the table indicates that more data exchange does have some contribution to the final performance. We should do some trade-off between higher performance and better communication efficiency.

Meanwhile, comparing with data transmit the 100 %, FedAvg always has less accuracy performance proving that the two steps aggregation HCA could work like a boundary to regular aggregation orientation. It could protect the contributing features in local and learn the outstanding characteristics from other clients.

5 Conclusion

We presenting a fancy framework for Federated Learning not only to reduce the communication consumption but also to solve the non-i.i.d problem. The Experiments on different datasets and network proves that FedMT is a feasible way to solve two severe problems at the same time. We achieve the competitive results in less communication and better performance comparing with FedAvg. We hope our work could imply in practical circumstances and build a better safety and efficiency world.

6 Future Work

Our work right now is preliminary in solving the non-i.i.d situation. This work is aim to solving the bandwidth overwhelm by transmitting partial filters. As the results show that partial transmission has the potentiality to be smaller and more precisely. However, the manual configuration limits the optimization of the whole work. We want to build a framework which will build the filter masks automatically. The new framework can optimize the workflow of making masks and cutting better boundaries by building a better aggregation regulation system. It may pick up more at start of training for faster converge and less at end of training for reducing exchange data and enhancing generalization. Meanwhile, the contribution calculation could not well indicate the importance of the filter. We will try more to find a better contribution value.

References

[1] General data protection regulation. `https://en.wikipedia.org/wiki/General_Data_Protection_Regulation`.

[2] Shupeng Gui, Haotao Wang, Chen Yu, Haichuan Yang, Zhangyang Wang, and Ji Liu. Adversarially trained model compression: When robustness meets efficiency. *CoRR*, abs/1902.03538, 2019.

[3] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015.

[4] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *CoRR*, abs/1808.06866, 2018.

[5] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[6] Brendan McMahan and Research Scientists Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`, 2017.

[7] Monica Ribero and Haris Vikalo. Communication-efficient federated learning via optimal client sampling, 2020.

[8] Stephen Boyd Stephen P. Boyd. *Distributed optimization and statistical learning via the alternating direction method of multipliers*.

[9] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *CoRR*, abs/1902.04885, 2019.

A FedMT Diagram

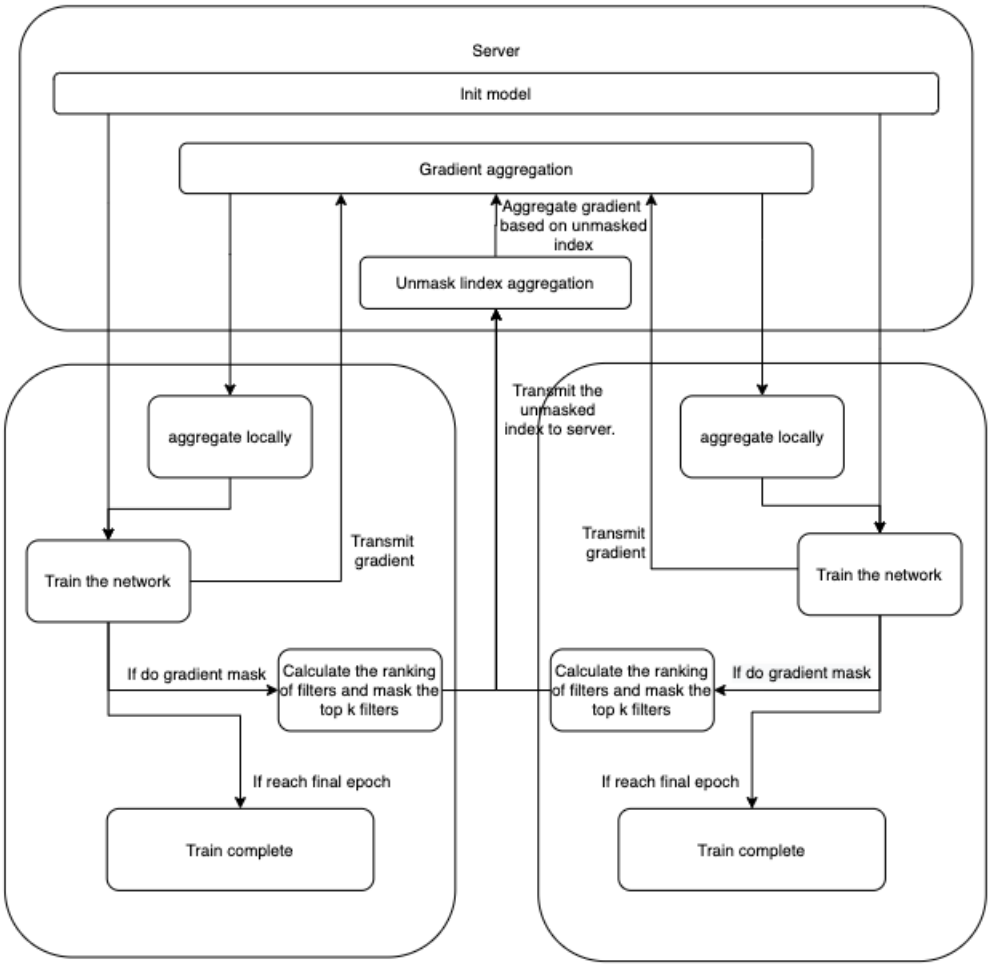

Figure 1: Fed Diagram for two clients