Dynamic Sampling and Selective Masking for Communication-Efficient Federated Learning

Shaoxiong Ji[§⋆], Wenqi Jiang[‡⋆] Anwar Walid[†] Xue Li[∓]
[§]Aalto University, Finland. Email: shaoxiong.ji@aalto.fi
[‡]Columbia University, USA. Email: wenqi.j@columbia.edu
[†]Nokia Bell Labs, USA. Email: anwar.walid@nokia-bell-labs.com
[∓]The University of Queensland, Australia. Email: xueli@itee.uq.edu.au
[⋆] First two authors contribute equally.

Abstract

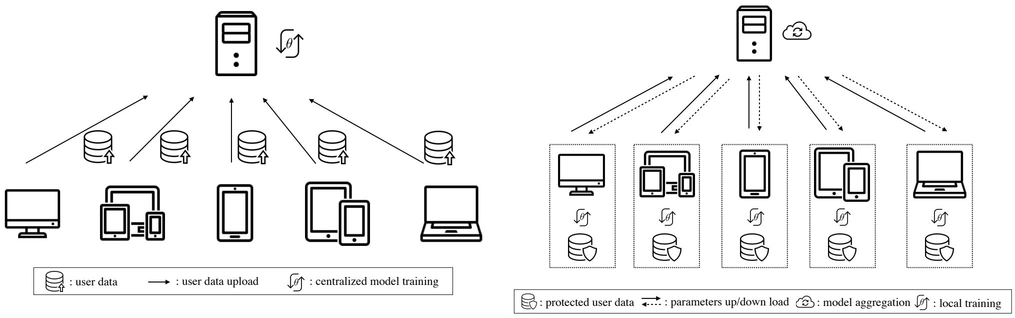
Federated learning (FL) is a novel machine learning setting which enables on-device intelligence via decentralized training and federated optimization. The rapid development of deep neural networks facilitates the learning techniques for modeling complex problems and emerges into federated deep learning under the federated setting. However, the tremendous amount of model parameters burdens the communication network with a high load of transportation. This paper introduces two approaches for improving communication efficiency by dynamic sampling and top-k selective masking. The former controls the fraction of selected client models dynamically, while the latter selects parameters with top-k largest values of difference for federated updating. Experiments on convolutional image classification and recurrent language modeling are conducted on three public datasets to show the effectiveness of our proposed methods.

Keywords— Federated Learning, Dynamic Sampling, Selective Masking

1 Introduction

Recent advances of machine learning techniques have drawn attention from many researchers and have boosted many industrial applications. However, traditional machine learning requires to collect massive data from users and trains a centralized model for prediction. To solve large-scale learning problem, distributed machine learning [26] is proposed for training in a distributed manner by allocating learning process in multiple computing nodes. Among popular distributed machine learning techniques, parameter server [19] is a typical one to scale up learning algorithms. With the widespread application of mobile communication technology and personal mobile devices, machine learning moves to edge devices and makes distributed agents more intelligent [12].

The increasing computational capacity of edge devices brings on-device computation into a reality. Observing privacy needs of mobile devices and unbalanced and non-IID nature of data in multi-party devices, federated learning is proposed to learn from decentralized data [22]. Unlike centralized learning, FL trains model without directly accessing private user data. By local computing and secure parameters transport, it can solve the security and privacy issues in the traditional centralized training. In this new setting, users' personal data are stored securely on their own devices. Conventional centralized learning and federated learning are illustrated in Fig. 1a and Fig. 1b respectively. To enable more efficient privacy preserving, federated learning can also be integrated with differential privacy [1] and secure multiparty computation [21]. It has also been applied in many fields like recommendation [4], computational phenotyping [14], and personalized next-word suggestion [11].



(a) The framework of centralized learning with data collection

(b) The framework of federated learning from decentralized data

Figure 1: The federated learning framework versus centralized learning framework. (a) Centralized learning with data collection. User data are uploaded to a central server to train a centralized model. (b) Federated learning. User data are stored securely on the user's devices to train a local model. Only the parameters from the local models are uploaded to the server for aggregation.

Federated deep learning takes deep neural networks as learning models on local devices and uses averaged model aggregation over sampled clients under the control of a central server. However, the humongous amount of transport cost in federated deep learning is one bottleneck for real-world application because deep learning models have a large number of parameters. Modern devices use wired communication or wireless communication with limited bandwidth. Thus, it is hard for current network transportation to handle such large amount of transportation, which makes communication-efficient federated learning a critical mission.

To increase communication efficiency, there are two strategies, i.e., to reduce the number of communication rounds between server and clients during federated training and to transport less parameters for each federated iteration. To save the transportation cost, current work [22] uses static sampling to select a fraction of client models, while other improvements [2] use compression algorithms for more efficient communication. In this paper, we improve the vanilla federated averaging methods by dynamic sampling for client models and selective masking on neural parameters of client models. The former one aims to reduce communication rounds, while the latter one aims to sample a fraction of parameters for transportation. During model downloads and uploads, our proposed methods can also be combined with cutting-edge compression algorithms for furthering communication efficiency.

Our contributions are summarized in the following three folds.

- We propose dynamic sampling strategy for federated averaging with an exponential annealing of sampling rate.
- We propose selective masking on neural parameters of client models to save the amount of data during model transportation.
- Through experiments on both convolutional image classification task on two popular image datasets and recurrent language modeling task on a typical text dataset, our proposed methods outperform their baseline methods in most cases.

This paper is organized as follows. A brief literature review is conducted in Sec. 2. In Sec. 3 and 4, a preliminary introduction of federated averaging algorithm and our proposed methods are presented respectively. A series of experiments are conducted in Sec. 5, together with a comprehensive comparison. Finally, we draw a conclusion in Sec. 6.

2 Related Work

We review the literature of federated learning from three aspects, i.e., federated learning itself, methods for communication-efficient federated learning, and real-world applications in the federated scenario.

2.1 Federated Learning

Federated learning is a decentralized machine learning method, which uses distributed training on local users without sending the data to a central server, firstly proposed by McMahan et al. [22]. A recent survey on federated machine learning [33] categorized federated learning into three aspects of vertical federated learning, horizontal federated learning and federated transfer learning.

Several improvements on the federated framework have been proposed. Geyer et al. [6] applied differential privacy preserving techniques on the user side to further privacy protection. To solve the bias of client models, Mohri et al. [24] proposed agnostic federated learning. Peterson et al. [27] integrated federated setting with per-user domain adaptation. Aggregation strategy is a key component of federated learning. To improve vanilla federated averaging, Ji et al. [11] proposed attentive federated learning for private neural language modeling. Lalitha et al [17] proposed to learning aggregation over a network of nodes. Besides, multiple distributed learning task can be taken as a multi-task learning problem. Smith et al. [14] proposed federated multi-task learning with distributed primal-dual optimization.

2.2 Communication Efficiency

Communication efficiency is considered as an important evaluation metric of federated learning. To enable efficient communication, Konečnỳ et al. [15] proposed sketched updating method for reducing transport cost in federated learning and use random subsampling in their sketched updating method. Bonawitz et al. [3] designed a scalable federated learning system architecture. In the field of distributed machine learning, there are also several methods for saving communication cost. For instance, Alistrach et al. [2] proposed gradient quantization and encoding. Wen et al. [32] proposed ternary gradients. Sattler et al. [30] proposed sparse ternary compression.

2.3 Federated Applications

Federated learning has also emerged in many real-world application fields. For example, a federated meta learning framework is proposed by Chen et al. [4] for privacy preserving recommendation. Kim et al. [14] used federated learning for clinical concepts discovery from electronic health records using tensor factorization. Popov et al. [28] proposed to learn language models from distributed private data by fine-tuning. Other applications include personalized social photo sharing prediction [13], proactive social care in private communities [10], neuroimage analysis [20], and spam message detection [7].

3 Preliminaries

This section introduces some preliminaries of federated learning, including federated averaging algorithm and two strategies to save communication cost, i.e., static sampling on clients for federated aggregation and random masking on model parameters for saving uploading cost.

3.1 Federated Averaging

Federated averaging performs distributed training and learns efficiently from decentralized data using iterative averaging. Without accessing personal data, it can achieve the goal of preserving privacy with multi-party computation and differential privacy. Federated learning is highly preferred when data is highly sensitive, for example, medical data and financial credit data.

In the federated setting, there is a central server for model aggregation and a set of client devices for local training. One typical algorithm is federated averaging where the central server averages client models to obtain a global model that can well generalize distributed clients. For deep neural networks, the global model at t is denoted as $\Theta_t = \{W_t^1, \ldots, W_t^i, \ldots\}$ with multiple layers, where a real-value matrix $W_t^i \in \mathbb{R}^{d_1 \times d_2}$ represents parameters in the i-th layer. The parameters of global model are then distributed to clients via network transportation. For each selected client, the downloaded model is trained on its physical device using its own data, with the trained local model in the i-th client at t-th communication round denoted as Θ_t^i. Then, clients uploads trained models to the central server for model aggregation by computing the weighted average. Given totally m selected clients as a set S in the distributed environment, the aggregated global model is computed in Eq. 1.

$$\Theta_{t+1} = \frac{1}{m} \sum_{i \in S}^{m} \Theta_t^i \tag{1}$$

For each layer in the neural network model, pair-wise matrix summation is calculated. To consider data imbalance, federated averaging performs weighted averaging by taking number of training samples as weights. The aggregation of client models is calculated as

$$\Theta_{t+1} = \frac{1}{m} \sum_{i \in S}^{m} \frac{n_i}{n} \Theta_t^i, \tag{2}$$

where n_i is the number of training instances in the i-th device and $n = \sum_{i=0}^{m} n_i$ is the total number of training samples of all selected clients.

3.2 Static Sampling

Federated learning applies static sampling by selecting a random fraction of clients for federated averaging. The server initially sets a sampling rate C, then it waits for updates from clients. Once there are enough updates to meet the sampling rate, the server will stop receiving updates and turn to federated averaging. During this procedure, the sampling rate C remains unchanged, and this is why it is so-called static sampling. The algorithm of federated averaging with static sampling is shown in Algorithm 1, which is executed on the central server.

Algorithm 1 Federated Averaging with Static Sampling

$C \in \mathbb{R}$ is a constant of the sampling fraction of clients; $R \in \mathbb{N}^+$ is the value of preset communication rounds
Input: a set of M registered clients $S = \{s_1, \ldots, s_M\}$
Output: updated global parameters Θ_{t+1}
procedure FEDERATED AVERAGING(S) ▷ Run on a central server
    Initialize model Θ_0, set the fraction of clients C, a set of client parameters $\{\Theta_0^1, \ldots, \Theta_0^{C*M}\}$
    for $t = 1 : R$ **do**
        Initialize an empty list L
        Number of sampled clients $m = \max(C * M, 1)$
        while len(L) < m **do** ▷ Static sampling
            Send connection request to clients
            if ACK from i-th client **then**
                call ClientUpdate(i, Θ_t)
                $\Theta_t^k \leftarrow receiveParam(i)$
                $L.add(\Theta_t^i)$
        $\Theta_{t+1} = \frac{1}{m} \sum_{i \in C}^{m} \frac{n_i}{n} \Theta_t^i$

3.2.1 Random Masking

In distributed computing environment, the bottleneck of federated learning is the huge amount of transport cost because the bandwidth of central server is fixed when receiving updates from distributed clients. To save the transportation cost, one naive strategy is to randomly select a part of updated model parameters for transportation. We term this as random masking because a random mask is applied into parameters of each neural layer. The algorithm of random masking is shown in Algorithm 2. Given a random seed in each client, random masking is randomized by $randi$ function which generates a matrix of $A \in \mathbb{R}^{a\times b}$ with γ of ones. A certain proportion of parameters is masked via pair-wise product of A matrix and parameters in each layer. Then, masked model is compressed when uploaded to the central server.

Algorithm 2 On-Device Training with Random Masking

γ is the proportion of masked parameters; B is the local mini-batch size; E is the number of local epochs; η is the learning rate.
Input: ordinal of user k, user data X.
Output: updated user parameters Θ_{t+1} at $t + 1$.
procedure CLIENT UPDATE(k, Θ) ▷ Run on the k-th client
    B ← $(split\ user\ data\ X\ into\ batches)$
    for each local epoch i from 1 to E **do**
        for batch b ∈ B **do**
            $\Theta_{t+1} \leftarrow \Theta_t - \eta\nabla L(\Theta_t)$
    for each layer i from 1 to l **do**
        $(a,\ b) = shape(W^i_{t+1})$
        $A = randi([0\ 1],\ a,\ b,\ \gamma)$
        $W^i_{t+1} = A \otimes W^i_{t+1}$
    send Θ_{t+1} to server

4 Proposed Methods

We propose two strategies to save the communication cost. They are dynamic sampling and selective masking. The iterative model aggregation method uses federated averaging [22]. Dynamic sampling is proposed to control the sampling rate of client models, while selective masking chooses parameters according to the absolute difference value of parameters with a preset proportion γ and saves communication cost as well.

4.1 Dynamic Sampling

Static subsampling uses a fixed subsample rate through the whole training process, no matter what the epochs and training steps are. This method is easy for implementation by evenly choosing the number of clients to enable model aggregation. We propose dynamic sampling with a high sampling rate at first and then decrease the sampling rate during each communication. Our motivation is to accelerate convergence at the beginning of federated learning by involving more clients for model aggregation at the very beginning. Once a more generalized federated model is trained based on the initialization, our method dynamically decreases the number of clients for model aggregation to save communication cost. Even though it costs more at the beginning of federated training, its selected number of clients model declines rapidly after several rounds of training. The declining rate of sampling rate can be chosen accordingly to ensure that the total amount of parameter transportation of dynamic sampling is fewer than its counterpart of static sampling after certain rounds of communication.

Our proposed dynamic subsampling method uses exponential decay rate to anneal the sampling rate in the training process, where the subsample rate R is a function of current epoch t and decay coefficient

β as shown in Eq. 3.

$$R(t, \beta) = \frac{1}{\exp(\beta t)} \tag{3}$$

With the decreasing rate multiplied with a preset initial sample rate C, we get the dynamic sampling rate as $c = \frac{C}{\exp(\beta t)}$ at t-th training round. With the increase of communication rounds, sampling rate becomes very small, making less than one client selected for model aggregation. In practice, the minimum number of selected client models is set to two. Integrated with federated averaging, the algorithm of the dynamic sampling is written in Algorithm 3. The core difference between static sampling is the dynamically changed sampling rate.

Algorithm 3 Federated Averaging with Dynamic Sampling

$C \in \mathbb{R}$ is a constant of initial sampling rate; $c \in \mathbb{R}$ is a variable of the sampling fraction of clients; $R \in \mathbb{N}^+$ is the value of preset communication rounds; $k \in \mathbb{R}$ is a constant of the decay coefficient in Eq. 3.
Input: a set of M registered clients $S = \{s_1, \ldots, s_M\}$
Output: updated global parameters Θ_{t+1}
procedure FEDERATED AVERAGING(S) ▷ Run on a central server
    Initialize model Θ_0, set the the decay efficiency of sampling β.
    for $t = 1 : R$ **do**
        Initialize an empty list L
        Calculate sample rate $c = \frac{C}{\exp(\beta t)}$
        Number of sampled clients $m = \max(c * M, 1)$
        while len(L) < m **do** ▷ Dynamic sampling
            Send connection request to clients
            if ACK from i-th client **then**
                call ClientUpdate(i, Θ_t)
                $\Theta_t^k \leftarrow receiveParam(i)$
                $L.add(\Theta_t^i)$
        $\Theta_{t+1} = \frac{1}{m} \sum_{i \in C}^{m} \frac{n_i}{n} \Theta_t^i$

4.2 Selective Masking

This subsection turns to another improvement - selective masking technique. Model parameters stored in client devices take up the most of transportation cost. When transporting data from user devices to central servers, the random masking algorithm randomly selects some computed updates and discard the rest. However, this method is less heuristic because it is unable to select prior updates. Consequently, some important updates can be discarded when randomly select parameters.

We propose a selective masking to consider the importance of model parameters in each local training. Given a static masking rate on the proportion of model parameters as the selective criteria, only model parameters with the largest absolute difference are selected proportionally for federated aggregation and model updating. The principle of selective making is simple as illustrated in Fig. 2. First, the difference of current model parameters of i-th layer and updated counterpart in the next time step is calculated as

$$D_t^i = \mid W_{t+1}^i - W_t^i \mid \tag{4}$$

Given the masking proportion of γ, top-k largest values are selected together with their indices, where k equals to γ multiplied with the number of elements of weight matrix. With the selected indices, a mask matrix is generated as M which contains $1 - \gamma$ zeros and γ ones. Then, the masked weight matrix is calculated by pair-wise product of mask matrix and full weight matrix as

$$W_{t+1}^i = M \otimes W_{t+1}^i. \tag{5}$$

Finally, the masked parameters are compressed and ready for transportation via client-server communication.

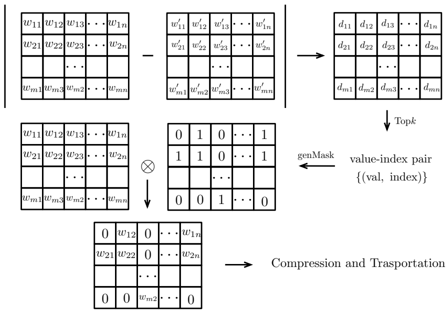


Figure 2: An illustration of selective masking

The algorithm of on-device training with selective masking is written in Algorithm 4, where the genMask function generates the mask matrix according to the top-k indices.

Algorithm 4 On-Device Training with Selective Masking

γ is the proportion of masked parameters; B is the local mini-batch size; E is the number of local epochs; η is the learning rate.
Input: ordinal of user k, user data X.
Output: updated user parameters Θ_{t+1} at $t+1$.
procedure CLIENT UPDATE(k, Θ) ▷ Run on the k-th client
    B $\leftarrow$ $(split\ user\ data\ X\ into\ batches)$
    for each local epoch i from 1 to E **do**
        for batch b $\in$ B **do**
            $\Theta_{t+1} \leftarrow \Theta_t - \eta \nabla L(\Theta_t)$
    for each layer i from 1 to l **do**
        $(a,\ b) = shape(W^i_{t+1})$
        $D = \mid W^i_{t+1} - W^i_t \mid$
        $indices, values = \text{topk}(D,\ \gamma)$
        $M = genMask([0\ 1],\ a,\ b,\ indices)$
        $W^i_{t+1} = M \otimes W^i_{t+1}$
    send Θ_{t+1} to server

5 Experiments

To study the performance of our proposed methods, experiments on serval datasets with different settings are conducted in this section. The proposed dynamic sampling and selective masking are compared with the baseline methods of static sampling and random sampling respectively.

5.1 Datasets and Settings

5.1.1 Datasets

We perform several neural architectures on two typical tasks - image classification and language modeling - on two image datasets of MNIST [18] and CIFAR-10 [16], and a natural language dataset of WikiText-

2 [23]. They are typical public datasets for training machine learning algorithms. MNIST[1] and CIFAR-10[2] are two widely used datasets with hand-written digits and objects respectively. And WikiText-2[3] is a dataset for word-level language modeling, which contains more than 2 million tokens extracted from Wikipedia. Our experiments are performed on these three widely-used datasets for validation. Statistics of these three datasets are summarized as in Table 1.

Table 1: A summary of datasets

Dataset	Type	# train	# test
MNIST [18]	image	60,000	10,000
CIFAR-10 [16]	image	50,000	10,000
Wikitext-2 [23]	token	2,088,628	245,569

5.1.2 Data Partitioning

Those three datasets are designed for training a centralized machine learning model. To make it suit for federated learning setting, data partitioning is adopted for generating decentralized datasets by sampling the whole dataset under independent and identical distribution (I.I.D.). We followed the data partitioning rule of MNIST and CIFAR-10 proposed by McMahan et al. [22]. Each partitioned subset of the whole dataset is regarded as a private dataset in a client device. The same partitioning rule is then applied to WikiText-2 to construct the I.I.D. federated dataset.

5.1.3 Settings

Federated learning involves with communication between distributed clients and the central server. However, due to the limitation of computing resources, we only apply the federated learning settings in a single Linux server to simulate the real scenario, which ignores the communication noise and delay in network.

We conduct both convolutional and recurrent neural networks according to specific tasks. For the image classification task, we use LeNet [18] as client model, and for furthering the effect of large-scale model, VGG [31] is implemented as client model on CIFAR-10 dataset. The language modeling task deals with sequential text data. Thus, we use long short-term memory networks (LSTM) [8] to capture sequential dependency with both tied and untied word embedding [9, 29]. All neural models are implemented by the PyTorch framework [25] and accelerated by Nvidia K40m GPU.

Our aim is to enable communication-efficient federated learning. Thus, transportation cost is considered for evaluation, which is related to sampling rate, masking rate and communication round. Taking a single communication between one client and the server with full model parameters as unit, it is calculated as

$$f(\beta, \gamma) = \frac{\gamma}{R} \sum_{t=1}^{R} \frac{C}{\exp(\beta t)} \tag{6}$$

for R rounds of client-server communication. In the following section, we take transportation cost together with prediction accuracy as evaluation metrics.

5.2 Convolutional Image Classification

We first begin with the task of convolutional image classification in the federated setting. Two groups of comparison are conducted on static versus dynamic sampling and random versus selective masking

[1]Available at `http://yann.lecun.com/exdb/mnist/`

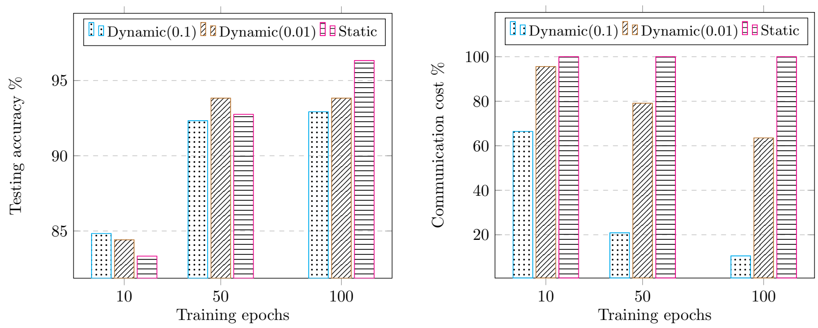
[2]`https://www.cs.toronto.edu/~kriz/cifar.html`

[3]`https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-2-v1.zip`

on MNIST digits classification using LeNet. Then, analogical comparison is further perform using large-scale VGG model on CIFAR-10 objects classification. Due to the sample rate decay, the dynamic sampling method can train more federated rounds than static method given a same transport cost and the same initial sampling rate. For example, with a decay coefficient of 0.1 and the same amount of transportation cost, the dynamic method can update 31 epochs, while static method can only train 10 epoch of updates. In the following experimental analysis, convolutional image classification task is performed under separate comparison first and then with two methods combined together for federated training.

5.2.1 Static Versus Dynamic Sampling

Sampling strategies are conducted in static and dynamic manners. We take the whole cohort of clients as initial training. The static sampling method keeps this rate during the whole process of federated training, while the dynamic counterpart takes an exponential decay on the sampling rate with decay coefficient in Eq. 3 of 0.01 and 0.1. Results of prediction accuracy and communication cost are reported in Fig. 3 after 10, 50 and 100 rounds of federated training. According to Fig. 3a, with the increase of training epochs, prediction accuracy grows steadily for all three settings. For less rounds of federated training (that is 10 and 50 rounds), testing accuracy of static sampling is poorer than dynamic sampling with 0.01 as decay coefficient. But when the sampling rate drops faster (i.e., a higher decay coefficient of 0.1), testing accuracy of dynamic sampling is imparted. After 50 and 100 round of training, static sampling gains a better prediction performance than dynamic sampling method. As for the transportation cost during federated communication, static method takes 100% of transportation, while the dynamic sampling method can efficiently save communication cost. With the increase of training epochs and the decay coefficient, much more rounds of client-server communication have been saved as illustrated in Fig. 3b.



(a) Prediction accuracy (b) Communication cost

Figure 3: Static versus dynamic sampling with 100% clients for initial model aggregation on MNIST dataset. For dynamic sampling, decay coefficient is set as 0.01 and 0.1.

5.2.2 Random Versus Selective Masking

After that, we evaluate the performance of selective method using top-k masking. In this section, we fix the sampling rate to be 0.1, and conduct experiments with random masking and selective masking. For fair comparison, these two methods use the same hyper parameter setting for 10 rounds training, and the learning rate is set to 0.01. Experimental results are reported in Fig. 4 where the masking rate varies from 0.1 to 0.9. With a relatively higher masking rate, the testing accuracy of random masking and selective masking is close. Selective masking performs a bit higher testing accuracy for masking rate of 0.8 and 0.9. When a large number of parameters are discarded with the rate of 0.1 and 0.2, the performance of random masking drops dramatically. According to this result, our proposed top-k selective masking method can remain a stable performance to save communication cost even with a high proportion of parameters ignored.

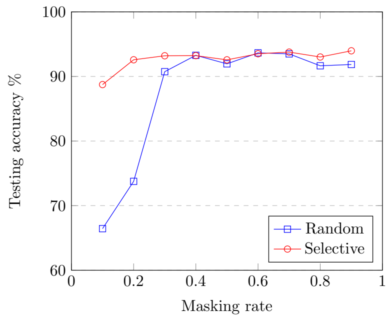
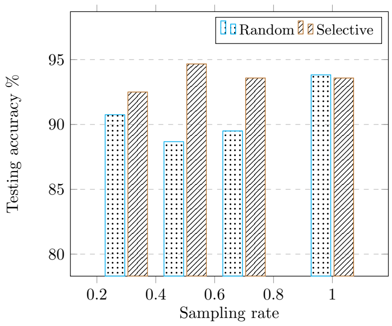


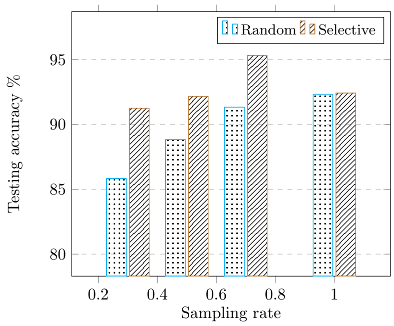
Figure 4: Random masking versus selective masking with static sampling rate of 0.1 for 10 rounds federated training on MNIST dataset.

5.2.3 Combined Experiment

After separate comparison of two proposed methods, we then combine them together into federated training for evaluation. In this section, four initial sampling rates of 0.3, 0.5, 0.7, and 1.0 are included for dynamic sampling method. As for the selective masking, two decay coefficients of 0.01 and 0.1 are used. Experimental results after 50 training rounds are shown in the two bar charts of Fig. 5. Selective masking outperforms random masking in dynamic sampling setting of these two cases except when initial sampling rate equals 1 with the decay coefficient of 0.01.



(a) Decay coefficient of 0.01



(b) Decay coefficient of 0.1

Figure 5: Random masking verses selective masking with dynamic sampling using sample coefficient of 0.01 and 0.1 after 50 training epochs on MNIST dataset.

5.2.4 Experiments on CIFAR-10

Then, we further experiments on CIFAR-10 dataset using VGG-16 model to evaluate the performance on large-scale model. The aggregated prediction accuracy of random and selective masking after 100 federated training rounds is shown in Fig. 6 where static sampling with 100% sampling rate is applied. Notice that our aim is to compare the performance of masking methods but not to the performance of image classification model. Thus, we do not fine tune the client learner. And all the experiments haven't achieved the state-of-the-art results of centralized training as the comparison is conducted in the federated setting with limited communication rounds. But our comparison is reported with the same setting of client learner to ensure fair comparison of two proposed methods and their counterparts. As we can see from that table, top-k selective masking method outperforms random masking for masking rate from 0.1 to 0.6. When the masking rate is high, these two methods gain similar testing accuracy. Our proposed top-k selective masking can remain satisfactory performance with large proportion of parameters saved in federated training.

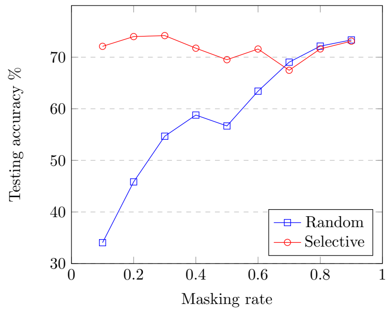
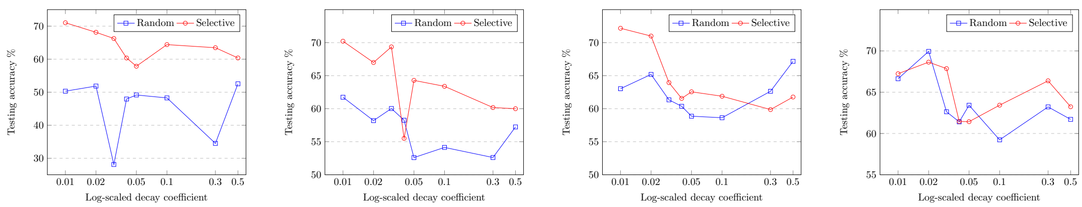


Figure 6: Aggregated prediction accuracy using random masking and selective masking with VGG model on CIFAR-10 after 100 federated training rounds.

In addition, we conduct experiments with both of sampling and masking strategies adopted for deeper analysis. This comparison aims to evaluate the effect of decay coefficient in dynamic sampling with masked updating applied. The results of using masking rates of 0.3, 0.5, 0.7 and 0.9 are reported in Fig. 7 where the x-axis is log-scaled. These figures show that when masking rate is 0.3, selective masking outperforms random masking with all settings of decay coefficient. For masking rate being 0.5 and 0.7, selective masking gains better testing accuracy in most cases. With a higher masking rate of 0.9, the performance gap of these two methods is narrow. Generally, with a larger decay coefficient (more communication-efficient), the performance experiences a fluctuation and decreases to a relatively low level when the decay coefficient set as 0.5.



(a) Masking rate = 0.3 (b) Masking rate = 0.5 (c) Masking rate = 0.7 (d) Masking rate = 0.9

Figure 7: The effect of different decay coefficients on dynamic sampling with federated aggregation using different masking rates on CIFAR-10.

5.3 Recurrent Language Modeling

Mobile keyboard suggestion with private neural language modeling is a typical application of federated learning, which has interaction with users to provide supervised labels. In this section, we model the next word prediction in mobile keyboard as private RNN-based language modeling. Specifically, we adopted gated recurrent unit (GRU) [5] as the client learner. GRU is a simplified variant of the long short-term memory (LSTM) network [8] which suits for saving communication cost with less parameters. Natural language corpus usually has a large vocabulary. To furthering communication-efficient federated learning, tying word embedding and word classifier [9, 29] is introduced by using shared parameters. In this section's experiments, tried embedding is applied. For the evaluation metric, we use the aggregated perplexity. Perplexity is a standard metric for language modeling task. According to its definition, lower perplexity means better performance.

We firstly compared the effect of sampling strategies. Fig. 8 takes 50 rounds of client-server communication with different rates of masking to compare the performance of static and dynamic sampling. This bar chart shows that dynamic sampling achieves a lower perplexity in most cases, excluding $\beta = 0.5$ with masking rate of 0.5 and 0.7, and $\beta = 0.1$ with masking rate of 0.8 and 0.9.

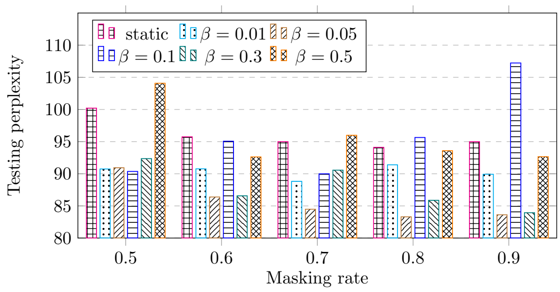


Figure 8: Static versus dynamic sampling with masked updating using different masking rate after 50 communication rounds on WikiText-2.

Secondly, we compared random masking with selective masking. Results using different masking rates are reported in Fig. 9. Our proposed selective masking is better for larger masking rates. Surprisingly, random masking gains better performance when masking rate is low. It is hard to interpret why random masking is better even when a large number of parameters are discarded for updating. One possible guess is that the randomness improves the generalization of aggregated recurrent model, making the testing perplexity decrease.

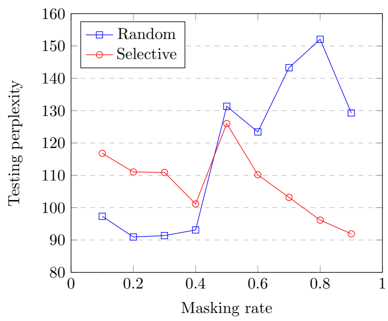


Figure 9: Random versus selective masking with different masking rates on WikiText-2.

5.4 Discussion

Comprehensive experiments on single computing node are conducted to mimic the federated setting in this section. Two tasks of convolutional image classification and recurrent language modeling are performed with comparison analysis. Our proposed achieves competitive performance in most experimental settings, which provides empirical approaches for saving communication cost in federated setting. For simplicity, we ignore the loss of network transmission. But admittedly, real-world environment of federated setting is more complicated which requires further simulation experiments. Deep neural models have a large number of parameters. Taking VGG-16 as an example, its total number of parameters is more than one hundred million. High-speed network techniques are also required. Due to the lack of computing resources, we leave experimental simulation on multiple computing nodes for the future work.

6 Conclusion

Federated learning decouples modeling training and data accessing, which protects data privacy. But it incurs communication cost issue when it combines with large-scale deep neural networks. To enable communication efficient federated learning, this paper proposes two empirical approaches - dynamic sampling and selective masking to save communication cost while ensures satisfying prediction performance. The proposed two strategies can save the number of server-client communication and save the amount of model parameters for each transmission. Experiments on convolutional image classification and recurrent language modeling show that our proposed methods gain competitive results.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[4] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[6] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[7] Hongmei He, Tim Watson, Carsten Maple, Jörn Mehnen, and Ashutosh Tiwari. A new semantic attribute deep learning with a linguistic attribute hierarchy for spam detection. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3862–3869. IEEE, 2017.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

[10] Shaoxiong Ji, Guodong Long, Shirui Pan, Tianqing Zhu, Jing Jiang, Sen Wang, and Xue Li. Knowledge transferring via model aggregation for online social care. *arXiv preprint arXiv:1905.07665*, 2019.

[11] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[12] Jing Jiang, Shaoxiong Ji, and Guodong Long. Decentralized knowledge acquisition for mobile internet applications. *World Wide Web*, 2020.

[13] Eunice Kim, Jung-Ah Lee, Yongjun Sung, and Sejung Marina Choi. Predicting selfie-posting behavior on social networking sites: An extension of theory of planned behavior. *Computers in Human Behavior*, 62:116–123, 2016.

[14] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Federated tensor factorization for computational phenotyping. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 887–895. ACM, 2017.

[15] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[17] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[20] Xiaoxiao Li, Yufeng Gu, Nicha Dvornek, Lawrence Staib, Pamela Ventola, and James S. Duncan. Multi-site fmri analysis using privacy-preserving federated learning and domain adaptation: Abide results. *arXiv preprint arXiv:2001.05647*, 2020.

[21] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

[22] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[23] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[24] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS Workshop*, 2017.

[26] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[27] Daniel Peterson, Pallika Kanani, and Virendra J. Marathe. Private federated learning with domain adaptation. *arXiv preprint arXiv:1912.06733*, 2019.

[28] Vadim Popov, Mikhail Kudinov, Irina Piontkovskaya, Petr Vytovtov, and Alex Nevidomsky. Distributed fine-tuning of language models on private data. In *International Conference on Learning Representation (ICLR)*, 2018.

[29] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163, 2017.

[30] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *arXiv preprint arXiv:1903.02891*, 2019.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519, 2017.

[33] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.