SmartIdx: Reducing Communication Cost in Federated Learning by Exploiting the CNNs Structures

Donglei Wu[1], **Xiangyu Zou**[1], **Shuyu Zhang**[1], **Haoyu Jin**[1], **Wen Xia**[1,2*], **Binxing Fang**[1]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[2]Peng Cheng Laboratory, China
{donglei.wu, xiangyu.zou, shuyu.zhang97, haoyu.jin97}@hotmail.com, xiawen@hit.edu.cn, fangbx@cae.cn

Abstract

Top-k sparsification method is popular and powerful for reducing the communication cost in Federated Learning (FL). However, according to our experimental observation, it spends most of the total communication cost on the index of the selected parameters (i.e., their position information), which is inefficient for FL training. To solve this problem, we propose a FL compression algorithm for convolutional neural networks (CNNs), called SmartIdx, by extending the traditional Top-k largest variation selection strategy into the convolution-kernel-based selection, to reduce the proportion of the index in the overall communication cost and thus achieve a high compression ratio. The basic idea of SmartIdx is to improve the 1:1 proportion relationship between the value and index of the parameters to n:1, by regarding the convolution kernel as the basic selecting unit in parameter selection, which can potentially deliver more information to the parameter server under the limited network traffic. To this end, a set of rules are designed for judging which kernel should be selected and the corresponding packaging strategies are also proposed for further improving the compression ratio. Experiments on mainstream CNNs and datasets show that our proposed SmartIdx performs $2.5\times-69.2\times$ higher compression ratio than the state-of-the-art FL compression algorithms without degrading training performance.

Introduction

Federated Learning (denoted as **FL**) (McMahan et al. 2017; Konecný et al. 2016b) allows many clients to collaboratively train a model while keeping their private datasets in their local, and has become a popular privacy-preserving machine learning approach (Wei et al. 2019; Zheng et al. 2020; Ng et al. 2020; Liu et al. 2020a,b; Smith et al. 2017). Broadly speaking, the main processes of FL can be described as below: ① Clients download the current global model from the central parameter server. ② Clients train the downloaded global model on their private datasets in parallel, and generate a new local model. ③ Clients upload their new local model to the parameter server. ④ The server aggregates the received local models, and generates a new global model as the result of the current training round. When the server sends back the averaged global model to clients, and clients begin the next training round by continually training the received model with their private datasets. These four steps are iteratively performed between clients and server until the global reaches the required accuracy.

One of FL's mainstream scenarios is that the server locates in a data center with good Internet resources, and clients are edge devices (e.g., smartphones, IoTs, or wearable devices) (Konecný et al. 2016a,b; McMahan et al. 2017; Sattler et al. 2020; Smith et al. 2017; Zheng et al. 2020) which are limited in low bandwidth and unstable network. Existing studies show that residential Internet connections tend to reach far higher download than upload speeds (Goga and Teixeira 2012; Konecný et al. 2016a; Kairouz et al. 2019; Cui et al. 2020; Zhang et al. 2021a, 2022), thus there exist a serious bottleneck in FL when a huge number of clients send their local model to the server, especially when the uploaded models are the complicated and large-scale convolutional neural networks (CNNs).

To alleviate the communication bottleneck of FL, a natural idea is to compress the network traffic by reducing the total size of uploaded model parameters. Recently, Top-k sparsification-based compression techniques significantly reduce communication overheads of federated/distributed learning without sacrificing the training convergence speed and inference accuracy (Dryden et al. 2016; Strom 2015; Sattler et al. 2020; Sattler et al. 2019; Aji and Heafield 2017; Lin et al. 2018; Chen et al. 2018; Abdi and Fekri 2020; Zhang et al. 2020). In these approaches, instead of transmitting the whole local model, weight updates or gradients (denoted as parameters) with the largest variations after local training will be selected and uploaded to the parameters server for reducing the network traffic. Here the selected parameters are regarded as individuals and they are usually organized in pairs (e.g., $<$*the value*, *the index*$>$ of a selected parameter). The index is used to indicate a selected parameter's position in the model (i.e., which parameter is selected), and the value is the specific numerical value of this parameter. In the server, the value will be used to update the previous global model's parameters at the corresponding position according to the index. Thus only the value participants in model training while the index does not.

Many previous works attempt to employ quantization techniques (Aji and Heafield 2017; Dryden et al. 2016; Wen et al. 2017) and encoding schemes (Sattler et al. 2020; Strom

*Corresponding author.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2015; Sattler et al. 2019) to further reduce the sizes of the '*value*' and '*index*' parts, respectively. However, according to **our** 1^{st} **key observation**, '*the value*' part is greatly compressed ($16\times-32\times$) while '*the index*' part is hard to be compressed ($2\times-3\times$), which thus leads to the ***'index' part takes an increasingly higher share of the network traffic***. For example, according to our experimental observation, there are about $89.4\%-94.1\%$ traffic costs spent on transmitting '*the index*' parts after using quantization and encoding techniques in Top-k sparsification based FL compression methods (Aji and Heafield 2017; Sattler et al. 2019; Sattler et al. 2020; Strom 2015; Zhang et al. 2021b).

On the other hand, previous work (Chen et al. 2018) mentioned that the traditional Top-k parameter selecting strategy that selects parameters with a large variation, may miss some critically important parameters with a small variation. These parameters could be also related to high activity input features. Thus we aim at exploring new communication-efficient selecting strategies in this paper. By studying the convolution kernels structures in CNNs, we try to design a **'Kernel-based' parameter selecting strategy**, in which the convolution kernels are regarded as a whole selecting unit (i.e., all the parameters in the convolution kernel are selected). Meanwhile, we find **the** 2^{nd} **key observation that when transmitting the same amount of parameters, this selecting parameters with the whole unit of convolution kernel achieves approximate training performance compared with the traditional Top-k strategy in FL** (detailed in Section). More importantly, by exploiting the structures of convolution kernel in CNN models, we can let parameters in a convolution kernel share an index (i.e., one *index* for multiple *values*) after using this 'kernel-based' parameter selecting strategy, which significantly reduces the proportion of the index part. **From another perspective, kernel-based selection strategy delivers more updated value to the server than traditional Top-k sparsification with the same communication cost, which is beneficial to prompting the FL training.**

In this work, motivated by the above two key observations, we propose a novel FL compression algorithm (called SmartIdx) to significantly reduce communication cost in FL, which optimizes the 1:1 proportion of selected parameter pairs (i.e., $<$*the value*, *the index*$>$) to n:1 by regarding the convolution kernel as the parameter selecting unit in convolutional layers. More specifically, there are three steps for SmartIdx: ① Selecting Parameters: several 'important' parameters of a CNN model are selected according to their variations, which are used to select 'important' convolution kernels. ② Collecting Properties: the $Size$ and $Number$ properties of each selected parameter are recorded, which are used to determine the encoding strategy of convolution kernel candidates. ③ Packaging Parameter: combined with the collected properties, several rules are designed to encode the final uploaded parameters into three kinds of packages (i.e., Individual Package, Kernel Package, and Pattern Package) for transmission. In the encoded kernel/pattern packages, multiple parameters share one index to reduce the FL communication cost.

Generally, the contributions of this paper are four folds:

- We **observe** that there is a large proportion of communication cost spent on the index (i.e., indicating which parameters are selected) in mainstream Top-k sparsification based FL compression methods, which is useless for the model learning in FL. Thus the index cost is expected to be reduced for improving the FL compression ratio and training performance.
- Meanwhile, we experimentally **verify** that when transmitting the same amount of parameters (i.e., with the same sparse ratio), our proposed 'Kernel-based' selection strategy that regards parameters in the convolution kernel as a whole selecting unit, achieves an approximate Federated Learning training performance to the traditional 'Top-k' strategy that selects individual parameters.
- Based on the kernel-based selection strategy, we **design** a novel FL compression algorithm called SmartIdx. It allows ① parameters in a convolution kernel share one index, which can significantly reduce the index proportion in the total traffic of FL communication. And ② the selected kernels with the same sign will share one sign, which can further reduce the value cost in the total traffic.
- Experiments on typical CNN structures and datasets suggest that our proposed SmartIdx achieves $2.5\times-69.2\times$ compression ratio when transmitting the same amount of parameters, and achieves higher test accuracy with the same network traffic limitation, than the state of arts.

Background and Related Work

Federated Learning (FL) is a particular case of distributed learning, which aims at casting off the collection of the private data from participants. However, the huge number of clients, the limited network resource of each client, and the increased size of the model make the communication problem becomes a bottleneck in practical FL.

To reduce the communication cost of FL, FedAvg algorithm proposed by (McMahan et al. 2017) to enlarge the communication interval, thus offloads the expensive bandwidth resource to the local computation. Based on FedAvg, sparsification, quantization, and encoding techniques are further exploited to reduce the size of the model weight updates/gradients communicated between server and client. Broadly speaking, the sparsification strategy reduces the network overhead by communicating sparse local model that only include k 'important' parameters (Chen et al. 2018; Lin et al. 2018). Quantization and encoding schemes can reduce the size of numerical value and position information of these k 'important' parameters (Seide et al. 2014; Wen et al. 2017; Alistarh et al. 2017; Sattler et al. 2020; Hu et al. 2020).

Typically, a sparse Binary compression (SBC) framework (Sattler et al. 2019) reduces the communication cost in FL. In SBC, all weight-updates but the fraction k with the highest magnitude will be set to zero (Top-k sparsification). Further, the sparse weight-updates are quantized to binary (quantization), the index of the non-zero elements will be encoded by optimal Golomb encoding (positions coding). Additionally, a Deep Gradients Compression (DGC) algorithm (Lin et al. 2018) uses Top-k sparsification to reduce the size of gradients, and run-length coding to com-

Approaches	Index	Value	σ
Scalable-DNN (Strom 2015)	$\sim$11-bit	1-bit	0.08
SBC (Sattler et al. 2019)	$\sim$8.4-bit	1-bit	0.11
Grad. Dropping (Aji and Heafield 2017)	32-bit	2-bit	0.03

Table 1: The average traffic composition of four gradient sparsification based Federated/Distributed Learning compression methods for each selected parameter.

press the traffic of sparse gradient's index. To ensure no loss of accuracy, DGC employs momentum correction and local gradient clipping on the Top-k sparsification. In addition, DGC also uses momentum factor masking and warmup training to overcome the staleness problem caused by the reduced communication. In Top-k sparsification based compression methods, a common and important step is to accumulate those unuploaded updates in local and add them to the next FL rounds, which has been proven to be essential to maintain the model performance during the training (Karimireddy et al. 2019; Alistarh et al. 2018; Stich, Cordonnier, and Jaggi 2018; Zheng, Huang, and Kwok 2019).

In this paper, we set the FedAvg algorithm as the baseline, and evaluate our proposed SmartIdx algorithm with the SBC and DGC in Section . More related work about FL compression approaches are introduced in supplementary[1].

Observation and Motivation

Observation 1. As already described, the steps of traditional Top-k sparsification-based methods (Aji and Heafield 2017; Sattler et al. 2020; Lin et al. 2018; Strom 2015), are: ① Sorting all parameters in the model according to their variations in a local training round. ② Selecting Top-k (i.e., the largest k) ones in the sorted results, regarding them as individuals (i.e., independent units), and recording them in pairs as $<$*the value*, *the index*$>$ of a selected parameter. Generally, the value and the index will be further compressed by quantization and coding technique in different methods. For the simplicity, assume that both the floating value and the integer index of each parameter need 32 bits to represent, we can define the compression ratio of value and index as:

$$CR_v = \frac{32k}{T_{Q(v)}}, \quad CR_i = \frac{32k}{T_{C(i)}} \tag{1}$$

k is the number of selected parameters with large variation. $T_{Q(v)}$ and $T_{C(i)}$ denote the size of the quantized value and the coded index, respectively. Existing compression approaches (Alistarh et al. 2017; Wen et al. 2017; Sattler et al. 2020) quantize the floating value to $1-2$ bits, which corresponding to the CR_v is about $16-32$. While the coding scheme (e.g., golomb coding) used to encode the integer index to 8-11 bits, which corresponding to the CR_i is about $2-3$ as suggested in approaches (Sattler et al. 2020; Sattler et al. 2019; Strom 2015). Obviously, CR_i **is much lower than** CR_v**, which causes the index to still takes a huge share of the total traffic after compression** in these approaches. Compared with '*the value*' part will be used to update global model in the central server, '*the index*' part is only employed to indicate which parameters are selected. Excessive communication cost on index means a low efficiency in FL communication. In this paper, we define the transmitting efficiency σ for FL compression methods:

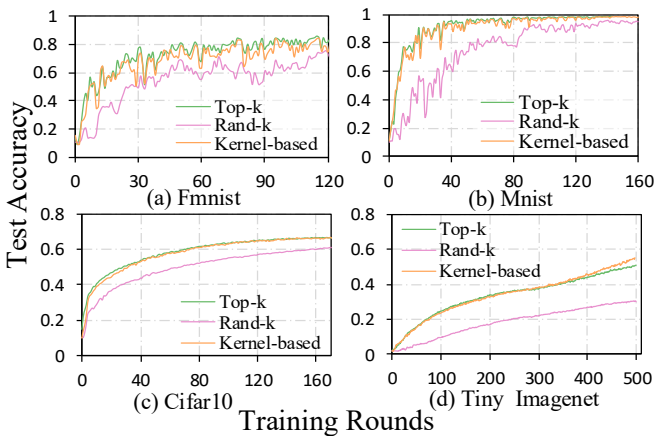


Figure 1: Comparisons of inference accuracy among three selecting strategies for FL compression.

$$\sigma = \frac{K}{T_v + T_i} \tag{2}$$

T_v and T_i are communication cost (bit) of value and index, K denotes the amount of uploaded parameters. Table **??** shows the communication cost composition and corresponding σ of 4 typical Federated/Distributed Learning compression approaches, according to their detailed design methodologies. We find that most of the traffics are spent on transmitting the index, but not the value, and σ of these approaches also quite low.

Observation 2. Top-k sparsification-based compression techniques achieve very promising performance in FL, and the theoretical analysis of the convergence has been proven by several existing studies (Alistarh et al. 2018; Karimireddy et al. 2019; Zheng, Huang, and Kwok 2019; Stich, Cordonnier, and Jaggi 2018), but the shortcoming about index cost in 1^{st} observation inspire us to explore a communication-efficient parameter selection strategy.

It is well known that parameters in a convolution kernel always work together, and kernels are often used as a monolith to extract features in CNN structures. More importantly, we notice that this structured property helps to reduce the index proportion because only one index is needed to determine all parameter's position in the same selected convolution kernel. Following this observation, we design a parameter collection strategy by regarding parameters in the convolution kernel as a whole selecting unit for sparsification. In doing so, the index proportion in the total traffic is reduced as the numbers of the index are dramatically decreased.

Intuitively, the training performance (i.e., convergence rate and test accuracy) of kernel-based selection should be approximate to the Top-k selection as they both consider the L_{inf} norm as a measure of 'importance', otherwise any a block of continuous parameters are selected also need only

[1]https://github.com/wudonglei99/smartidx

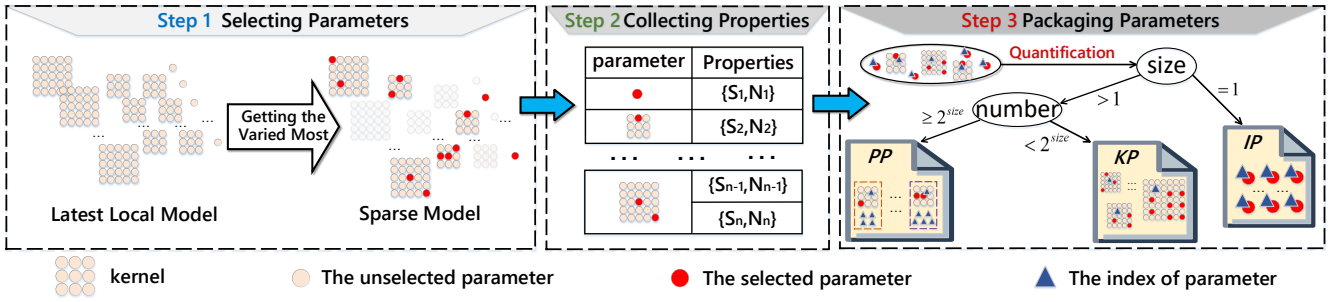


Figure 2: General workflow of SmartIdx. *IP*, *KP*, and *PP* are short for individual package, kernel package, and pattern package.

one index and reduce the communication cost. Thus, the experiment of training performance is designed to empirically prove the reasonableness of kernel-based selection. Specifically, we employ three parameters selecting strategies (i.e., Top-k, Rand-k, Kernel-based) to reduce the communication cost in FL, and compare the training performance among these methods (as shown in Figure 1).

- '*Top-k*' denotes that each client selects ***K*** parameters that are varied most for each training round;
- '*Rand-k*' denotes that each client randomly selects ***K*** parameters for each training round;
- '*Kernel-based*' (method of this paper) denotes that each client selects several convolution kernels where k varied-most parameters locate in. And there are total ***K*** parameters included (the detailed selection algorithm will be detailed in Section).

Three approaches use the same sparse ratio (SR), which is defined as the proportion of selected parameters per round. The close blue and green curves in Figure 1 suggests that '*Kernel-based*' performs an approximate FL model training results as '*Top-k*', which are significantly better than the *Rand-k* selection strategy. These experimental results are consistent with our expectation (More experiments of different CNNs are shown in supplementary[1]). Essentially, the *kernel-based* selection and *Top-k* selection both try to pick up some 'important' parameters in each round, and their difference is the selection granularity. *Top-k* regards a single weight as the basic unit while the *kernel-based* uses a convolution kernel as the basic unit.

In a nutshell, the above two key observations motivate us to propose an efficient FL compression algorithm that focuses on reducing the indexing cost by using 'Kernel-based' selection strategy, which will be detailed in the next section.

Design Methodologies

Overview

In this paper, we propose SmartIdx, an efficient FL compression approach that uses the 'kernel-based' selection strategy to reduce index cost while maintaining the FL training efficiency. As shown in Figure 2, SmartIdx has three key steps:

1. **Selecting Parameters**: Selecting these 'important' parameters of CNNs, i.e., those varied most in each FL training round, which will guide SmartIdx to further select the 'important' convolution kernels.
2. **Collecting Properties**: Acquiring and analyzing the selected parameters' two properties corresponding to the convolution kernels: *Size* and *Number*, which will be used as the clues for judgement and packaging of parameters in the next steps.
3. **Packaging Parameters**: Classifying the selected parameters (in convolutional layers or others) to get the selected convolution kernels and other selected individual parameters (e.g., locating in full-connection layers), which will then be encoded into three packages, including individual package, kernel package, and pattern package. In this step, our packaging method performs a higher transmitting efficiency by reducing the index proportion.

Selecting Parameters

As discussed in Section , traditional Top-k sparsification-based techniques select parameters according to their absolute values of variations or gradients (Aji and Heafield 2017; Sattler et al. 2020; Strom 2015; Lin et al. 2018; Chen et al. 2018). Similarly, our approach follows them to select several parameters for further 'Kernel-based' selection.

Generally, clients usually run multiple training iterations in a FL training round. Specifically, assuming that the i^{th} parameter ω_i in the local training is updated to ω_i' after a training round, the i^{th} parameter's variation degree is calculated as: $\Delta_i = |\omega_i' - \omega_i|$.

We calculate Δ values for all parameters after an FL training round, and then, the k largest Δ values will be identified to select k parameters. Note that the hyper-parameter k decides the sparse ratio and the final compression ratio, which will be discussed in Section .

Collecting Properties

In this step, we traverse the selected parameters and collect two properties of them:

- **Size** S: This records whether a selected parameter locates in a convolution kernel. If the selected parameter locates in a convolution kernel, S is configured as the size of the kernel; Otherwise, S is set to the value '1'.
- **Number** N: If a selected parameter locates in a convolution kernel, and the size of this convolution kernel is s,

N records the total number of convolution kernels whose sizes are s. If the parameter does not belong to a convolution kernel, N is set to the value '0'.

The two properties reflect where the selected parameters are located in CNNs and will be utilized for further compression as described in the next subsection.

Packaging Parameters

After collecting properties of the selected parameters, all the selected parameters and other parameters located in the related convolution kernel will be quantized: first averaged to the value μ according to their absolute values, and then simply represented by +μ or -μ (i.e., one bit each parameter along with their shared μ), as other Top-k sparsification-based FL compression methods do (Strom 2015; Sattler et al. 2020). Next, these quantized parameters will be packaged as described in Algorithm 1.

Specifically, the selected and quantized parameters will be encoded into three packages:

- A parameter w_i will be encoded into the **Individual Package (IP)** if $S_i = 1$, which means it locates in FC layer, bias layer, or just a 1×1 convolution kernel.
- A parameter w_i will be encoded into the **Kernel Package (KP)**, if it locates in a larger convolution kernel (i.e., $S_i \neq 1$), while there are less than 2^{S_i} convolution kernels in the CNN whose size is S_i.
- Otherwise, a parameter w_i will be encoded into **Pattern Package (PP)**, which means there are more than 2^{S_i} kernels in the CNN whose size is S_i.

Note that the **Pattern Package** can be considered as a further optimization on the **Kernel Package**, to achieve a higher compression ratio. More specifically, 2^S represents the number of all possible combinations of the S signs (i.e., patterns '+' or '-') for all parameters in the convolution kernel with a size of S. When there are more than 2^S kernels with a size of S, there must be at least two convolution kernels with the same pattern (i.e., the same signs '+' or '-').

For example, for the 3×3 convolution kernels, there are 2^9 patterns at most. If more than 2^9 kernels are with the size of 3×3, there must be at least two convolution kernels with the same pattern. In this case, SmartIdx can further reduce the repeated transmission for the convolution kernels with the same pattern by (1) classifying the corresponding convolution kernels (i.e., $N_i > 2^{S_i}$ and $S_i > 1$) according to their patterns; and then (2) for those kernels have the same pattern, only recording one pattern and multiple indices of the selected kernels which belong to this pattern.

According to the above Algorithm 1, the index and value costs are both significantly reduced since ① all the parameters in the same convolution kernel share a common index, and ② the cost for transmitting repeated signs of different kernels could be saved. Moreover, as suggested by the studies (Strom 2015; Sattler et al. 2020), instead of communicating the absolute non-zero positions, communicating the distances between all indices can further save the communication cost of the index, so we employ Golomb encoding (Golomb 1966) to compress the index of parameters in each package in SmartIdx.

Algorithm 1: Packaging the Selected Parameters.

Input: The selected parameters & their properties: w_i, $\{S_i, N_i\}$; // S_i: $Size$ of w_i located unit; N_i: $Number$ of w_i located unit; k_i is the convolution kernel containing w_i.
Output: Individual Package(IP), Kernel Package(kP), Pattern Package(PP)

for w_i and $\{S_i, N_i\}$, i = 0, 1, 2, . . . **do**
  if $S_i = 1$ **then**
    Individual Package ← w_i
  else if $S_i > 1$, $N_i < 2^{S_i}$ **then**
    Kernel Package ← k_i
  else if $S_i > 1$, $N_i > 2^{S_i}$ **then**
    Pattern Package ← k_i
  end if
end for
return IP,KP,PP

Communication and Computation Cost Discussion

Communication Cost: The communication costs of SmartIdx per round are composed of the cost of individual package C_{ip}, kernel package C_{kp}, pattern package C_{pp}, and metadata M:

$$C = \underbrace{W_{ip} + G(i_{ip})}_{C_{ip}} + \underbrace{\sum_{i=0}^{K} S_i n_i + G(i_{kp})}_{C_{kp}} + \underbrace{\sum_{j=0}^{P} S_j + G(i_{pp})}_{C_{pp}} + M \quad (3)$$

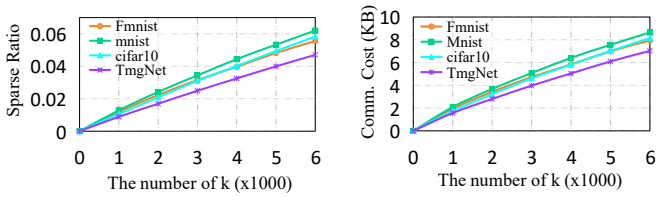
where G(·) denotes the communication cost of the golomb encoded index. W_{ip} denotes the number of parameters in *IP*. S_i and S_j denote the kernel size in *KP* and *PP*. In *kp*, n_i denotes the quantity of kernel with the size of S_i. Metadata *M* is used to mark some of the necessary information of each package, which only takes up a small part of traffic. Since all selected parameters are quantized to the mean of non-zero elements, we use 1 bit to record the sign of each value (i.e., 0 denotes negative value and 1 denotes positive value).

Computation Cost: As introduced in Section 4.1, there are mainly three steps in SmartIdx. Assume that the time complexity of Step 1 is O(n_1), where n_1 is the number of total parameters in the CNN. The time complexities of Step 2 and Step 3 are both O(n_2), where n_2 is the number of the selected parameters. Since $n_1 >> n_2$, the time complexity of SmartIdx is O(n_1), which is nearly the same as the traditional Top-k compression method.

Experimental Evaluation

Experimental Setup

Models and Datasets: Referring to existing studies (Caldas et al. 2018; Sattler et al. 2020; Konecný et al. 2016a; Li et al. 2020; Lin et al. 2018), the local models used for this work are structure and size simplified Vgg11 (Simonyan and Zisserman 2015), Resnet18 (He et al. 2016), and GoogleLeNet (Szegedy et al. 2015), which are named as vgg11*, resnet18*, GLnet*. The datasets used to training are IID (independent identically distributed) partitioned Tiny ImageNet (TmgNet) (Krizhevsky, Sutskever, and Hinton 2012)



(a) Sparse ratio with different k

(b) Communication cost with different k

Figure 3: The relationship between hyper-parameters k and (a) sparse ratio, (b) communication cost, with training differents CNNs on Cifar10.

/ Cifar10 (Krizhevsky and Hinton 2009), and Non-IID partitioned Mnist (LeCun 1998) / Fashion Mnist (FMnist) (Xiao, Rasul, and Vollgraf 2017). For the IID distribution, each client has the same amount of private data with various and random classes. For the Non-IID distribution, each client is assigned the same amount of two random classes of data. Additionally, each client has the same amount of data (See supplementary[1] for more illustrations).

Configuration of FL: In this paper, the base FL configurations are set by referring to the existed studies (Konecný et al. 2016a; McMahan et al. 2017), there are total of 100 clients, 10 clients will be randomly selected to participate in each FL round. Each client conducts their local model training by using Adam optimizer (Kingma and Ba 2015) for 1 epoch, the local batch size is set to 10.

Hyper-parameters: The only hyper-parameter of SmartIdx is the numbers of selected parameters k, which is described in Section . Note that in the SmartIdx, the sparse ratio and communication cost are not strictly proportional to the hyper-parameters k, they are two pairs of positively correlated variables. Considering that the client training GLnet* on different datasets. Figure 3 suggests that both the sparse ratio and communication cost have almost achieved linear growth with the rising of k. It means that the sparse ratio and communication cost in FL can be adjusted by k (results on more CNNs are shown in the supplementary[1]). Based on this property, next experimental evaluations could be conducted on different configurations of compression ratio and communication cost of SmartIdx.

Baseline and Compared Algorithms: In our experiments, the baseline is set to the non-compressed and full precise vanilla Federated Averaging algorithm (FedAvg). To evaluate the performance of SmartIdx, we compare the SmartIdx to 2 Top-k sparsification based FL comparison algorithms: SBC (Sattler et al. 2019) and DGC (Lin et al. 2018), which are introduced in Section . For certain CNN and dataset, we will obtain the baseline of converged test accuracy Acc_b by conducting the FedAvg with a certain round R_b.

Comparison of Compression Ratio with the Same Sparse Ratio

In this section, we compare SmartIdx with SBC and DGC in compression ratio with the same sparse ratio. As mentioned in Section , the sparse ratio of SmartIdx can be controlled by adjusting the hyper-parameters k. And there are similar hyper-parameters in SBC and DGC for controlling their sparse ratio (Sattler et al. 2019; Lin et al. 2018). According to their training performances upon various configurations of CNNs and datasets, we set a series of the same sparse ratios for each method to evaluate the communication costs (See supplementary[1] for more illustrations). In doing so, the communication costs of three methods with the same sparse ratio are obtained, and their *Compression Ratios (CR)* can be calculated by the size of the raw CNNs divided by the communication cost. Table **??** shows that the SmartIdx achieves $2.5\times - 69.2\times$ more compression ratio than SBC and DGC by reducing the traffic of index. These results suggest that SmartIdx greatly saves communication resources by utilizing the inherent n:1 relationship between index and value of the uploaded convolution kernels. And this superiority is favorable to the FL client with limited bandwidth resources.

CNNs	Methods	Fmnist	Mnist	Cifar10	TmgNet
VGG11*	SBC	42.0	43.7	39.7	40.0
	DGC	4.1	4.5	3.6	3.7
	SmartIdx	179.2	207.8	206.4	178.3
GLnet*	SBC	39.2	41.7	20.8	41.0
	DGC	4.2	3.9	2.4	3.8
	SmartIdx	113.8	109.3	72.4	101.8
Resnet18*	SBC	65.0	46.0	65.7	66.6
	DGC	6.5	5.3	6.3	6.5
	SmartIdx	413.9	367.0	422.9	413.0

Table 2: Compression ratio with the same sparse ratios of SBC, DGC, and SmartIdx on different CNNs and datasets.

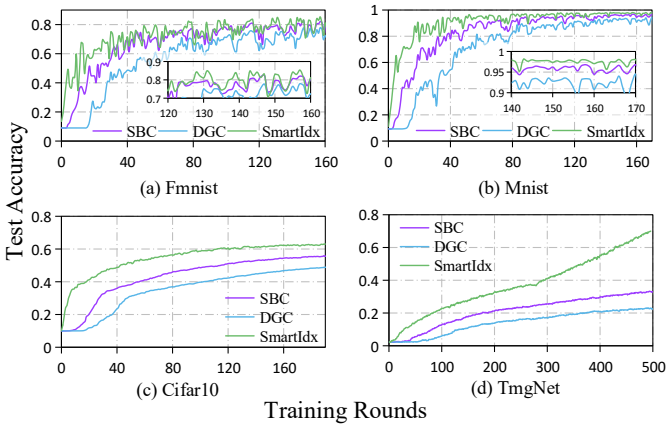


Figure 4: Training performance with the same communication costs of SBC, DGC, and SmartIdx on different datasets.

Convergence Rounds with the Same Communication Cost

As discussed in Section , sparsification- and quantization-based approaches (including our approach) reduce the traffic cost, but maybe decrease the valuable 'information' deliver, which incurs more training rounds to reach the required test

Datasets	CNNs	R_b/Acc$_b$	SBC				DGC				SmartIdx			
			CR	σ	acc	r	CR	σ	acc	r	CR	σ	acc	r
Cifar10	VGG11*	(200/68%)	38.55	1.94	68.01%	161	12.73	0.17	67.98%	199	206.39	9.60	68.24%	190
	GLNet*	(200/67%)	20.14	1.51	66.43%	198	3.80	0.17	67.01%	159	72.49	5.07	67.21%	194
	ResNet18*	(200/63%)	59.29	1.83	63.18%	172	11.66	0.17	63.08%	191	422.86	11.18	63.10%	186
TmgNet	VGG11*	(500/70%)	38.38	1.92	60.60%	500	9.02	0.17	68.90%	499	178.34	7.96	70.60%	474
	GLNet*	(500/55%)	40.10	1.80	45.18%	500	4.04	0.17	54.95%	499	101.81	4.47	55.21%	497
	ResNet18*	(500/70%)	54.29	1.76	60.17%	500	19.57	0.17	70.24%	448	413.05	10.63	70.00%	496

Table 3: Evaluations of compression ratio, transmitting efficiency, test accuracy and FL rounds on best practices.

accuracy and lower test accuracy. Thus there is a trade-off between the communication costs and training performance. In this section, we explore the training performance among SBC, DGC, and SmartIdx with **the same communication cost**. Similar to section , a series of communication costs of different configurations of CNN and dataset are set for SBC, DGC, and SmartIdx. With the same communication cost per round, we can compare the training performance of SBC, DGC, and SmartIdx.

Figure 4 shows the test accuracy curves of the FL global model on Resnet18* with 4 datasets. We learn that with the same communication cost per FL round, models trained by SmartIdx have higher test accuracy than SBC and DGC. Essentially, with the same communication cost, SmartIdx delivers more updates value to the server as most of the communication costs are spent on transmitting parameter's value. This superiority is favorable to FL clients who take full advantage of communication resource (See supplementary[1] for more results). In summary, compared with the SBC and DGC, SmartIdx reduces the communication cost with the same sparse ratio, or achieves better training performance with the same communication cost per round.

Comparison on Best Practices

In this subsection, we explore **best practices** among SBC, DGC, and SmartIdx. A best practice indicates that within a given FL communication rounds R_b, training to the target test accuracy Acc_b, the minimum sparse ratio SR_b (i.e., the maximum compression ratio) of a particular approach could obtain.

We evaluate three methods on 2 datasets and 3 CNNs, Table **??** shows best practices about compression ratio **CR**, transmitting efficiency σ, highest test accuracy **acc**, and training rounds **r** of them. Note that due to SBC and DGC cannot reach to the A_t within R_b in some cases, we select the results of these cases corresponding to the highest accuracy achieved by them. Table **??** suggests that the σ of SmartIdx is $2.5\times-65.8\times$ higher than SBC and DGC, which are consistent with our intuition in Section . As a result, SmartIdx achieves $2.5\times-36.2\times$ higher compression ratio than SBC and DGC on their best practices (More comparisons of best practices are shown in the supplementary[1]).

To further reveal the essence of the superiority of SmartIdx, we calculate the *bit number* consumed by **index** and **value** of SmartIdx in Individual Package (*IP*), Kernel Package (*KP*), and Pattern Package (*PP*), respectively. Additionally, **Sparse Ratio (SR)** and **Compression Gain (CG)** of each package are given. CG is calculated by the raw communication cost of a *selected parameter* divide by the communication cost of the compressed version of this parameter (i.e., the sum of coded index and quantized value), it reflects the final compression effect on a parameter. Figure 5 shows the details of best practices on Resnet18* and 4 datasets. (a) suggests that **the communication cost spent on the index is about** $12.1\times-31.7\times$ **that on the value in the (*IP*), which is similar to the traditional FL compression methods SBC and DGC**. In the *KP* and *PP*, due to parameters in a convolution kernel share an index makes the n:1 proportion between value and index, the communication costs spent on the index are greatly reduced. In Figure 5 (b), due to the convolution kernels belong to the same pattern share one signs in the pattern kernel, the communication costs spent on the value are further saved. As the result, the highest CG is achieved by *PP*, and the lowest CG is achieved *IP* in (c) (See supplementary[1] for results on more CNNs).

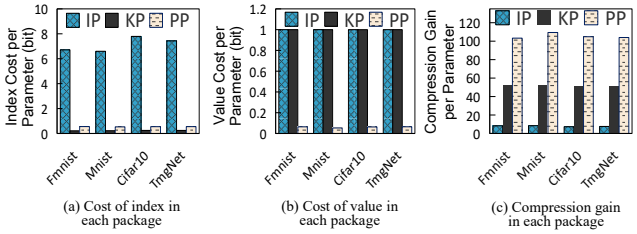


Figure 5: Comparisons of index cost, value cost, and compression gain among different packages.

Conclusion

In this paper, we observe the transmitting efficiency of the selected parameters in the communication of the existing Top-k sparsification methods in FL is quite low, which is because indices take a huge share of traffic costs. Therefore, we propose SmartIdx that uses a convolution-Kernel-based selection strategy by exploiting the structure of CNNs, which significantly reduces the index cost in the whole FL communication. Compared with the traditional Top-k sparsification methods, experiments show our approach achieves a much higher compression ratio without degrading the model convergence and accuracy of Federated Learning.

Acknowledgments

Thanks for anonymous reviewers' insightful comments and valuable suggestions. This research was partly supported by the National Key-Research and Development Program of China under Grant No. 2020YFB2104003; the National Natural Science Foundation of China under Grant no. 61972441; the Shenzhen Science and Technology Program under Grant no. JCYJ20200109113427092 and no. GXWD20201230155427003-20200821172511002; Guangdong Basic and Applied Basic Research Foundation No. 2021A1515012634.

References

Abdi, A.; and Fekri, F. 2020. Quantized Compressive Sampling of Stochastic Gradients for Efficient Communication in Distributed Deep Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 3105–3112. AAAI Press.

Aji; and Heafield. 2017. Sparse Communication for Distributed Gradient Descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 440–445.

Alistarh, D.; Grubic, D.; Li, J.; and et al. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *NeurIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, 1709–1720.

Alistarh, D.; Hoefler, T.; Johansson, M.; and et al. 2018. The Convergence of Sparsified Gradient Methods. In *NeurIPS 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 5977–5987.

Caldas, S.; Konecný, J.; McMahan, H. B.; and et al. 2018. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. arXiv:1812.07210.

Chen, C.; Choi, J.; Brand, D.; and et al. 2018. AdaComp : Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2827–2835.

Cui, L.; Su, X.; Zhou, Y.; and Zhang, L. 2020. ClusterGrad: Adaptive Gradient Compression by Clustering in Federated Learning. In *IEEE Global Communications Conference, GLOBECOM 2020, Virtual Event, Taiwan, December 7-11, 2020*, 1–7. IEEE.

Dryden, N.; Moon, T.; Jacobs, S. A.; and Essen, B. V. 2016. Communication Quantization for Data-Parallel Training of Deep Neural Networks. In *2nd Workshop on Machine Learning in HPC Environments, MLHPC@SC, Salt Lake City, UT, USA, November 14, 2016*, 1–8.

Goga, O.; and Teixeira, R. 2012. Speed Measurements of Residential Internet Access. In *Passive and Active Measurement - 13th International Conference, PAM 2012, Vienna, Austria, March 12-14th, 2012. Proceedings*, volume 7192 of *Lecture Notes in Computer Science*, 168–178. Springer.

Golomb, S. W. 1966. Run-length encodings. *IEEE Transactions on Information Theory*, 12(3): 399–401.

He, K.; Zhang, X.; Ren, S.; and et al. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. IEEE Computer Society.

Hu, Z.; Zou, X.; Xia, W.; and et al. 2020. Delta-DNN: Efficiently Compressing Deep Neural Networks via Exploiting Floats Similarity. In *ICPP 2020: 49th International Conference on Parallel Processing, Edmonton, AB, Canada, August 17-20, 2020*, 40:1–40:12. ACM.

Kairouz, P.; McMahan, H. B.; Avent, B.; ; and et al. 2019. Advances and Open Problems in Federated Learning. arXiv:1912.04977.

Karimireddy, S. P.; Rebjock, Q.; Stich, S. U.; and et al. 2019. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 3252–3261. PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Konecný, J.; McMahan, H.; Yu, F.; and et al. 2016a. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492.

Konecný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016b. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 1106–1114.

LeCun, Y. 1998. The MNIST Database of Handwritten Digits. http://yann.lecun.com/exdb/MNIST/. Accessed: 1998.

Li, T.; Sanjabi, M.; Beirami, A.; and et al. 2020. Fair Resource Allocation in Federated Learning. In *Proceedings of ICLR'20, Addis Ababa, Ethiopia, April 2020*.

Lin, Y.; Han, S.; Mao, H.; and et al. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *Proceedings of ICLR'18, Vancouver, Canada, April 2018*.

Liu, F.; Wu, X.; Ge, S.; and et al. 2020a. Federated Learning for Vision-and-Language Grounding Problems. In *Proceedings of AAAI'20, New York, NY, February, 2020*, 11572–11579.

Liu, Y.; Huang, A.; Luo, Y.; and et al. 2020b. FedVision: An Online Visual Object Detection Platform Powered by Federated Learning. In *Proceedings of AAAI'20, New York, NY, February, 2020*, 13172–13179.

McMahan, B.; Moore, E.; Ramage, D.; and et al. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of AISTATS'17, Fort Lauderdale, FL, April 2017*, volume 54, 1273–1282.

Ng, K.; Chen, Z.; Liu, Z.; and et al. 2020. A Multi-player Game for Studying Federated Learning Incentive Schemes. In *Proceedings of IJCAI'20, Yokohama, Japan, January 2021*, 5279–5281.

Sattler, F.; Wiedemann, S.; Müller, K.; and et al. 2019. Sparse Binary Compression: Towards Distributed Deep Learning with minimal Communication. In *Proceedings of IJCNN'19, Budapest, Hungary, July 2019*, 1–8.

Sattler, F.; Wiedemann, S.; Müller, K.; and et al. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9): 3400–3413.

Seide, F.; Fu, H.; Droppo, J.; and et al. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proceedings of INTERSPEECH'15, Singapore, September 2014*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated multi-task learning. In *NeurIPS 2017, Long Beach, CA, December 2017*, 4424–4434.

Stich, S. U.; Cordonnier, J.; and Jaggi, M. 2018. Sparsified SGD with Memory. In *NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 4452–4463.

Strom, N. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Proceedings of INTERSPEECH'15, Dresden, Germany, September, 2015*, 1488–1492.

Szegedy, C.; Liu, W.; Jia, Y.; and et al. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 1–9.

Wei, X.; Li, Q.; Liu, Y.; and et al. 2019. Multi-Agent Visualization for Explaining Federated Learning. In *Proceedings of IJCAI'19, Macao, China, August 2019*, 6572–6574.

Wen, W.; Xu, C.; Yan, F.; and et al. 2017. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In *Proceedings of NIPS'17, Long Beach, CA, December 2017*, 1509–1519.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747. .

Zhang, F.; Zhai, J.; Shen, X.; and et al. 2021a. TADOC: Text analytics directly on compression. *VLDB J.*, 30(2): 163–188.

Zhang, F.; Zhai, J.; Shen, X.; and et al. 2022. POCLib: A High-Performance Framework for Enabling Near Orthogonal Processing on Compression. *IEEE Trans. Parallel Distributed Syst.*, 33(2): 459–475.

Zhang, S.; Wu, D.; , H.; and et al. 2021b. QD-Compressor: a Quantization-based Delta Compression Framework for Deep Neural Networks. In *39th IEEE International Conference on Computer Design, ICCD 2021, Virtual Conferenc, October 24-27, 2021*. IEEE.

Zhang, X.; Zhu, X.; Wang, J.; and et al. 2020. Federated learning with adaptive communication compression under dynamic bandwidth and unreliable networks. *Information Sciences*, 540: 242–262.

Zheng, S.; Huang, Z.; and Kwok, J. T. 2019. Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback. In *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 11446–11456.

Zheng, W.; Yan, L.; Gou, C.; and et al. 2020. Federated Meta-Learning for Fraudulent Credit Card Detection. In *Proceedings of IJCAI'20, Yokohama, Japan, January 2021*, 4654–4660.