This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

Representative Kernels-based CNN for Faster Transmission in Federated Learning

Wei Li, Zichen Shen, Xiulong Liu*, Mingfeng Wang, Chao Ma, Chuntao Ding*, Jiannong Cao, *Fellow, IEEE*

Abstract—Federated Learning (FL) has attracted many attentions because of its ability to ensure data privacy and security. In FL, due to the contradiction between limited bandwidth and huge transmission parameters, it has been an ongoing challenge to reduce the model parameters that need to be transmitted to the server in the clients for fast transmission. Existing works that attempt to reduce the amount of transmitted parameters have limitations: 1) the reduced number of parameters is not significant; 2) the performance of the global model is limited. In this paper, we propose a novel method called **Fed-KGF** that significantly reduces the amount of model parameters that need to be transferred while improving the performance of the global model in FL. Since convolution kernels in a Convolution Neural Network (CNN) account for most parameters, our goal is to reduce the parameter transmission between the clients and the server by reducing the number of convolution kernels. Specifically, we construct an incomplete model with a few representative kernels, referred to the convolution kernels, that are solely updated during training. We propose Kernel Generation Function (KGF) to generate convolution kernels to render the incomplete model to be a complete one, and discard those generated kernels after training local models. The parameters that need to be transmitted only reside in the representative kernels, which are significantly reduced. Furthermore, there is a client-drift in the traditional FL because it adopts the averaging method, which hurts the global model performance. We innovatively select one or few modules (a module indicates a convolution function + several non-convolution functions) from all client models in a permutation way, and only aggregate the uploaded modules rather than averaging them in server to reduce client-drift, thereby improving the performance of the global model and further reducing the transmitted parameters. Experimental results on both non-Independent and Identically Distributed (non-IID) and IID scenarios for image classification and object detection tasks demonstrate that our **Fed-KGF** outperforms the SOTA methods. **Fed-KGF** achieves approximately 11% higher classification accuracy and roughly 33% fewer parameters than the recent FedCAMS model on CIFAR-10, and gains approximately 3.64% higher detection precision and around 37% fewer parameters than the SOTA SmartIdx model on COCO2017 datasets.

Index Terms—Federated learning, Convolution neural network, Representative kernels, Kernel generation function, Parameter reduction, Module selection.

1 Introduction

1.1 Problem Statement and Motivation

Federated Learning (FL) has been demonstrated as a powerful privacy-preserving distributed machine learning framework [1] [2] [3] [4] and applied to many real-world applications [5] [6] [7] [8]. In FL, all clients collaboratively train a global model under the orchestration of a central server by intensively transmitting all parameters of local models, rather than the data [1] [9]. This not only effectively models the data distributed across different clients, but also preserves the privacy. Note that training FL heavily relies on the parameter transmission among all clients and the server. However, the transmission cost is huge, due to the limited bandwidth and the enormous transmitted parameters as shown in Fig. 1. For example, Resnet18 [10] has around 11 million parameters, while YOLOX-L [11] holds 54.2 million parameters. If we have 10 clients and each client deploys a YOLOX-L, the total transmitted parameters are 54.2 $\times$ 10 = 542 million parameters at each epoch when adopting the traditional FL method (i.e., Fed-Avg). Assuming one epoch transmits 542 million parameters that spends one hour, 200 epochs would spend 200 hours. Although researchers attend to reduce the number of transmitted parameters, there are two challenges: 1) the reduced number of transmitted parameters is not significant; 2) the performance of the global model is limited after reducing the transmitted parameters. In deep neural network models based on convolution operations, the parameters of the convolution kernel usually account for a large proportion of the parameters within a network [12] (e.g., $\frac{the\ parameters\ of\ kernels}{the\ parameters\ of\ network} \approx 98.3\%$ in Resnet18). This motivates us to reduce the transmitted parameters between clients and the server by reducing the number of parameters of these kernels. In this way, we propose representative kernels-based approach to address the problem of parameter transmission for faster transmission

- *Wei Li and Zichen Shen are with the School of Artificial Intelligence and Computer Science & Engineering Research Center of Intelligent Technology for Healthcare, Ministry of Education & Jiangsu Key Laboratory of Media Design and Software Technology, Jiangnan University, Jiangsu, P. R. China. E-mail: cs_weili@jiangnan.edu.cn; 6223110033@stu.jiangnan.edu.cn.*
- *Xiulong Liu* is with the College of Intelligence and Computing, Tianjin University, P.R.China. Corresponding author. Email: xiulong_liu@tju.edu.cn.*
- *Mingfeng Wang is with the Department of Mechanical and Aerospace Engineering, Brunel University London, UB8 3PH London, U.K. E-mail: mingfeng.wang@brunel.ac.uk.*
- *Chao Ma is with the School of Cyber Science and Engineering, Wuhan University, Wuhan, P.R.China. Email: whmachao@ieee.org.*
- *Chuntao Ding* is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. Corresponding author. E-mail: chtding@bjtu.edu.cn.*
- *Jiannong Cao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P.R.China. Email: csjcao@comp.polyu.edu.hk.*

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

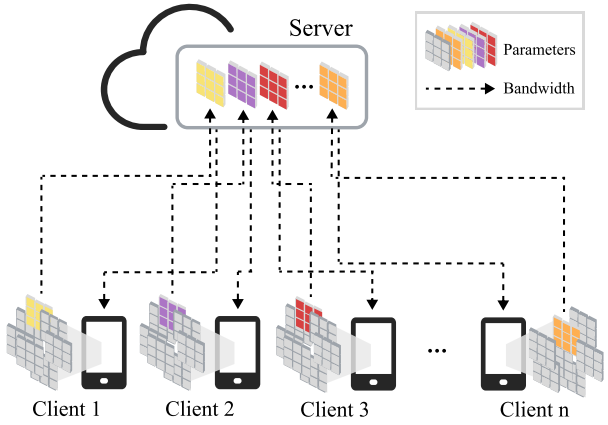


Fig. 1: The example of contradiction between huge transmitted parameters and the limited bandwidth.

while improving the performance of the global model.

1.2 Limitations of Prior Art

The SOTA works for reducing transmitted parameters in FL can be categorized into three classes: quantization [13] [14], sparsification [15] [16] [17], and quantization-sparsification methods. For the quantization method, it directly quantizes the model from higher precision to a lower one (e.g., from double format to integer one) before uploading the local updates to the server, which helps reduce parameters. However, the lower, even zero, precision will cause information loss, certainly hurting the performance of the global model, especially for the non-Independent and Identically Distributed (non-IID) data case [18]. For the sparsification method, it usually selects the absolute values of top-k of the gradients from each node, and then transmits these values in the full precision format [19]. Although it can reduce the parameters, the value of k is difficult to be determined, because different values cause various performance. Moreover, the sparse data require extra cost to calculate the indexes of top-k and then transmit these indexes, which help locate the sparse data in networks, so the transmission reduction is limited. The quantization-sparsification method [18] [19] tries to combine the two strategies to reduce parameters. It usually quantizes the top-k values after sparsification. Yet, the combination brings other challenges, such as difficult implementation [19] and higher transmission cost [18].

1.3 Our Approach

With the goal of reducing the amount of transmitted parameters between clients and the server while improving the global model performance, we propose **Fed-KGF** in this paper. First, we construct an incomplete network with a few base kernels. To take in inputs, we propose the Kernel Generation Function (KGF) to generate kernels for extracting the salient features of inputs. Although KGF is a commonly used concept and appeared in other studies as well, our function is different from other studies [20] [21] [22]. More details are discussed in Section 3. KGF consists of three components: 1) positivity and negativity. Since the values within kernels hold positivity and negativity [23], we should still keep the same positive and negative characteristics of extracted features from both generated and original base kernels in each layer [24]; 2) generated kernels. It renders the incomplete network to become a complete one to take in inputs and then extracts their features as the standard kernels, and this is achieved by index calculation with a random tensor [25]; 3) bias tensor. It avoids the zero case after index calculation. Second, we discard those generated kernels when transmitting the parameters from local models to the server, and call the rest kernels as representative kernels. Note that the generated kernels do not involve the updating, but the representative ones do, which means we only transmit the representative kernels between clients and the server. This significantly reduces the transmitted parameters. Since most existing FL models adopt the averaging method to calculate the parameters of the global model, there is the client-shit [26], causing the limited global model performance. Given that a CNN holds multiple stacked modules (a module is assembled by $Conv + Batch\_Norm + ReLu/Pooling$ functions) [27] [28], we innovatively propose a new permutation module uploading method, which randomly select one or few modules from a CNN, and then upload the selected modules across all clients to establish the global model in server. Because we only upload module to server, it further reduces the transmitted parameters. Besides, there is no averaging operation but solely assembles the uploaded modules to obtain the global model, thus avoiding the client-shift issue and then improving the global model performance. Last, we download the parameters from global model to each local one. After training, the global model generates kernels with KGF to perform the classification or object detection tasks.

1.4 Contributions and Advantages over Prior Work

Our innovations and contributions are shown as follows:

1) We propose a new transmitted parameter reduction strategy, which transmits a few of representative kernels between clients and server, in FL. To update the parameters of representative kernels, we present the Kernel Generation Function (KGF) to generate kernels to render an incomplete network being a complete one, and then train it with back-propagation in clients.
2) We propose a new module uploading method, which innovatively upload the random selected modules, rather than all modules as implemented in the traditional FL, from the network across all clients to the server in a permutation way, and then aggregate all uploaded modules to establish the global network. This further reduces the number of transmitted parameters, and avoids the issue of client-shift, thereby improving the performance of the global network.
3) We prove the effectiveness of proposed KGF and the convergence of **Fed-KGF** from the theoretical aspect. The experimental results show that the reduced parameters from **Fed-KGF** are reduced by approximately 36% on CIFAR10 and 33% on COCO2017 datasets, while **Fed-KGF** achieves higher performance than SOTA baselines from the empirical aspect.

The rest of this paper is organized as follows. Section 2 briefly reviews the FL variants in reducing transmitted parameters. Our **Fed-KGF** is introduced in Section 3, and

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

the experimental results are presented in Section 4. We conclude the paper in Section 5.

2 Related Work

In this section, we first introduce the federated learning, and then discuss the three categories of parameter reduction studies in FL according to their applied methods.

Federated Learning. Usually, the data in FL are stored locally in multiple clients [1] (e.g., mobile devices). To make the global model to learn the features of data, all clients collaboratively transmit their parameters to the server, rather than the stored data, to aggregate the parameters of the global model. After training, the global model has successfully extracted the data features from all clients and then towards performs different tasks (e.g., classification or object detection). The object of traditional FL is shown as follows.

$$\min_{\theta \in \mathcal{R}^d} f(\theta) \quad where \quad f(\theta) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \tag{1}$$

where $f_i(\theta) = \ell(x_i, y_i, \theta)$ indicates the loss of the prediction on the sample (x_i, y_i) with parameters θ of a network in i-th client. Since the traditional FL adopts Mathematical Average strategy, it is also called **FedAVG**.

Quantization. The quantization method usually transforms the format of parameters from float to integer. Since the integer format requires less storage space than the float format, researchers utilize quantization to reduce parameters for fast transmission in FL. Quantization Stochastic Gradient Descent (QSGD) [29] focuses on the quantization of gradients in each layer by normalizing each parameter and then rounds down each normalized value with $\frac{|v|}{L2}$ where $|v|$ denotes the absolute parameters and $L2$ denotes their normalized results. Obviously, QSGD suffers from information loss. Although there are other SGD quantization variants (e.g., Hyper-sphere Quantization [30] and Adaptive Quantization [31]), they still have the same issue and then degrades the performance of model. Federated Periodic Averaging and Quantization (FedPAQ) [13] claims that the transmission bottleneck is from the huge parameters of models within clients, so it renders partial clients to participate in each round of the training after quantizing their updates. Yet, the directions of gradients in partial locals might be inconsistent with that of gradients of the global model, the performance of the global model is unexpected, especially for the case that the data are distributed over all clients in the non-IID manner. Federated Averaging with Compression (FedCOM) [14] proposes the local gradient tracking scheme to address this issue. It tracks the quantized results of gradients and then to accumulate these results. After that, server receives the accumulated results and then averages them to obtain the estimated global gradients. FedAQ [32] can accelerate the updates during training, with an efficient quantization scheme. Nevertheless, they require the extra resources to store the processed information (e.g., accumulated gradients for FedCOM and auxiliary and original local parameters for FedAQ) and to calculate the gradients. Hence, larger bandwidth is required for transmission, with the lower performance of global model. There are other quantization methods (e.g., vector quantization [33] [34], minimization quantization errors [35], and lossy FL [36]). However, they either have compression distortion [33] [34] or still hold the non-negligible quantization errors [35].

Sparsification. The sparsification method reduces the parameters by removing the less important parameters in a network, because many parameters do not have much contribution to the performance of model [28]. Deep Gradient Compression (DGC) [37] finds only 0.1% gradients are of top importance in some cases. It employs momentum correction and local gradient clipping to make gradients sparse, which helps relieve the transmission bandwidth. However, DGC holds a drastic exchange and it requires extra bandwidth, so the relieved bandwidth is limited in practice. FetchSGD [16] utilizes the linear count sketch to compress the parameters of local models with top-k strategy, and then merges the compressed parameters of clients in server. It introduces the error accumulation to mitigate the information loss in the process of compression. Yet, selecting the appropriate parameters to figure out the range of the smallest quantization error and to determine the value of k are non-trivial tasks. The same challenge also exists other studies which adopt top-k to perform sparsification (e.g., rTop-k [38] and Top-k SparseSecAgg [39]). Communication-Efficient Adaptive Federated Learning (FedCAMS) [17] integrates the error feedback with the adaptive optimization to guarantee the convergence, in which the error feedback helps reduce compression error when utilizing top-k strategy to estimate the biased result. Although the adaptive strategy is suitable for the sparse case, it may suffer zero gradients, still remaining lower performance. Other accumulation-compression models (e.g., AdaComp [40] and RedSync [41]) also hold the similar challenges. SmartIdx [42] tries to address the inefficient transmission on the index of the selected parameters in top-k method. It extends the top-k largest variation selection strategy to the convolution-kernel-based selection, which helps relieve the overall transmission cost and thus achieves a higher compression ratio. Yet, such a compression might bring information loss and thus the lower performance. FedPM [43] adopts the *stochastic* binary mask to find the optimal sparse random network. Different from the traditional sparse training strategy, FedPM utilizes the *Sigmoid* function to update parameters of the binary mask, and each client samples a binary mask from the trained mask with Bernoulli sampling operation. Since Bernoulli sampling belongs to random sampling, parameters might be not sparsified, resulting in insignificant parameter reduction, given that the number of sampled parameters could be large during training. After uploading the mask to server, some parameters within global model are assigned by this mask and others are normalized to zero, thus causing a limited performance of the global model.

Quantization-Sparsification. This strategy attempts to combine the above two methods to transmit parameters in FL [40] [19] [18] [42]. Qsparse-local-SGD [19] provides mathematical proofs to analyze both convex and non-convex objectives for the distributed case with local computations, and then proposes a quantizer with sparsification and local computations using error compensation to mitigate the transmission bottleneck. However, these proofs need to satisfy rigorous hypotheses, which are often not cases in practice. Sparse Ternary Compression (STC) [18] casts

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

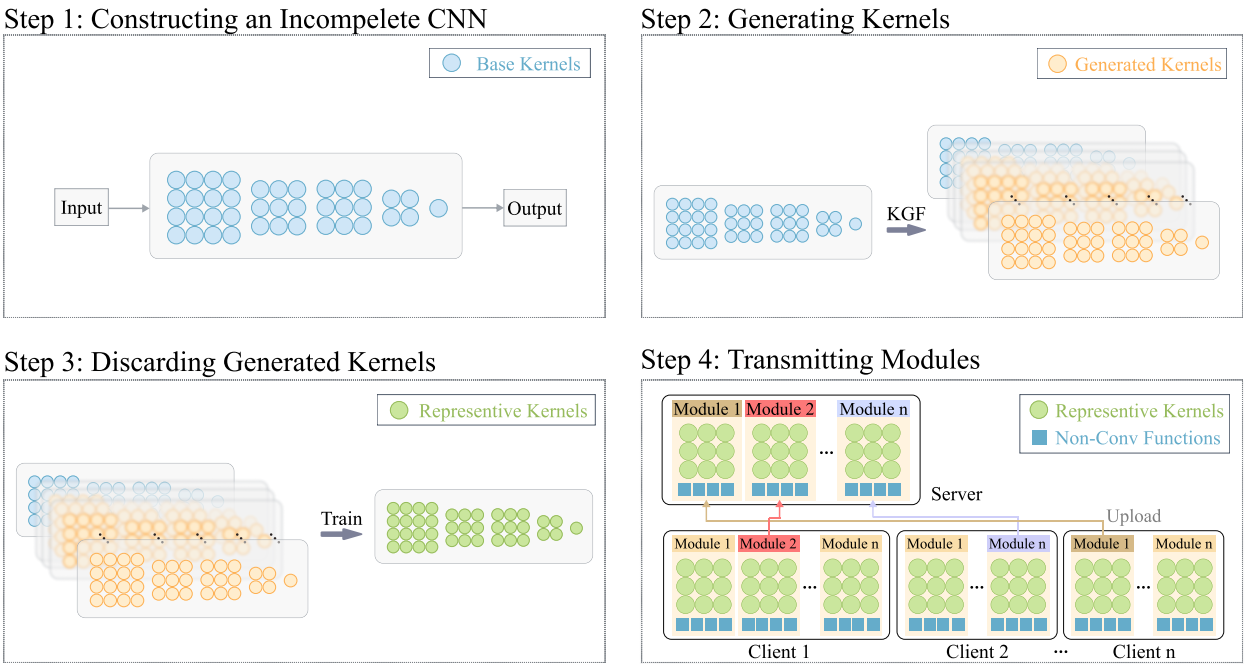


Fig. 2: The architecture of our **Fed-KGF**.

the light on the downstream transmission to reduce the cost of total transmission. It transmits the distances among the top-k values for relieving the transmission rather then transmitting the absolute positions of top-k in network, with Golomb encoding. Yet, Golomb encoding heavily relies on the Base-m, where the larger the value of m is, the longer length the Golomb encoding will be, which results in larger transmission cost. It is noted that the SOTA studies (e.g., SmartIdx [42]) have empirically shown that they outperform Quantization-Sparsification methods (e.g., STC [18] and AdaComp [40]), so this paper only compares the above two SOTA studies with **Fed-KGF**.

3 Fed-KGF

3.1 Overview

Fig. 2 shows the architecture of our **Fed-KGF**. In Step 1, we construct an incomplete network with a few kernels. To take in the input images, we propose the **Kernel Generation Function** (KGF) to generate kernels for extracting data features in each convolution layer in Step 2. We train this network in each client, and discard the generated kernels during back propagation in Step 3. With this, only representative kernels remain. To further reduce the parameters for faster transmission, we propose the permutation module uploading method, which selects one or few modules from this network in a permutation way, and then upload the selected modules across all clients to the server in Step 4. The server aggregates all uploaded modules to establish the global network, and then distributes its parameters to the local network of each client.

3.2 Design of Kernel Generation Function

In general, the traditional FL model deploys n full CNN models to n clients, and each client holds exactly one CNN model [1]. Clients transmit the parameters of all models to the server after updates. However, a full CNN model has a large number of parameters (e.g., Resnet18 with 11 million parameters and YOLOX-L with 54.2 million parameters). It absolutely results in the transmission burden. Since many parameters are redundant [44] [10] and kernels take most of parameters [12] [45], an intuitive idea is to reduce the parameters by limiting the number of kernels. Inspired by [24] [25], which have proved that kernels can be generated or decomposed with inexpensive cost, we focus on how to generate kernels in an incomplete CNN to take in inputs and to extract their features in this section.

As shown in Step 1 of Fig. 2, we first construct an incomplete network with a few kernels, and view these kernels as the base kernels. We denote a certain kernel as w, and i-th element within a kernel is named as w_i. Since the number of kernels is usually the exponent of 2, we predefine a ratio (i.e., $\frac{2^n}{2^m}$ with $n > m$ in which n indicates the number of base kernels and m is an integer) to set the number of base kernels. Then, we propose the Kernel Generation Function (KGF) to generate kernels defined as bellow.

$$w' = KGF(w) = sign(w)(|w|^{\beta} + \alpha) \tag{2}$$

where $sign(\cdot)$ operation keeps the positivity and negativity of extracted features from generated kernels as that from base kernels. β denotes a tensor, and it renders each entry within $|w|$ to perform exponential operation (i.e., $w_i^{\beta_i}$). All elements of β are from a random range, (e.g., [2, 10]). As to α, it is a bias tensor, with range from 0.00001 to 0.1. As can be seen from the above formula, our KGF includes three components: $sign(\cdot)$, $|w|^{\beta}$, and the bias α. The process of generating kernels is: 1) we "copy" the base kernels and the number of copied kernels is pre-defined; 2) Eq. (2) is utilized to render each entry within copied kernels to be different, then obtaining new kernels. In this way, the generated kernels are assembled into the incomplete network and then extract features of inputs as the standard kernels.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

Let w be a tensor with $k \times k$ size, it holds positive and negative values. Given $|w|^{\beta} + \alpha$, w' obviously holds the characteristic of positivity only. In such a scenario, the generated kernels cannot effectively extract the salient features of data, which certainly degrades the performance of the global network.

Proposition 1. Assuming w is a tensor with $k \times k$ size where it holds positive and negative values, and w' is also a tensor with $k \times k$ size and positivity only if removing $sign(w)$. Let x be the input data and $k = 2$, the performance of model is not satisfactory if w' loses the characteristics of positivity and negativity.

Proof. Under the condition of $k = 2$ within a certain layer, we assume $w = \begin{bmatrix} positive & negative \\ positive & positive \end{bmatrix}$, we still obtain the feature matrix which holds the same characteristics as w after performing $sign(w_i)(|w_i|^{\beta_i} + \alpha_i)$ one by one as defined in Eq. (2). In such a scenario, the characteristics of extracted features from generated kernels (i.e., w') are the same as that from original base kernels, w. This guarantees that the extracted features from both generated and original kernels have the similar characteristics, because the features extracted by the kernels which are from the same layer have similar characteristics [24]. If $w' = |w|^{\beta} + \alpha$, then the matrix could be $\begin{bmatrix} positive & positive \\ positive & positive \end{bmatrix}$. In such a scenario, the characteristics of extracted features from generated kernels (i.e., w') do not hold negativity. This indicates that w' can not effectively extract the salient data features, so it cannot also help improve the model performance. Hence, the performance of model is not satisfactory if w' loses the characteristics of positivity and negativity.

Since the parameters of network are usually initialized into the range of [-1, 1] in practice [28], the values of $|w|^{\beta}$ are different from those of w when β is from the range of [2, 10]. Considering an extreme scenario, $w = \begin{bmatrix} 0.1 & -0.2 \\ 0.01 & 0.2 \end{bmatrix}$ and $\beta = \begin{bmatrix} 10 & 8 \\ 9 & 10 \end{bmatrix}$, each entry in w' could be as small as negligible in practice after performing $|w|^{\beta}$. In other words, $w' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. Such kernels are not able to extract the salient features from inputs, and certainly hurts the performance of model. Therefore, it is necessary to avoid such a case.

Proposition 2. Let $|w|^{\beta}$ be $\begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$ and α be an random bias tensor, the generated kernels, w', are not equal to zero, and can extract the salient features.

Proof. Assuming the algorithm is at k-th iteration, we have $|w|^{\beta} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$ and $\alpha = \begin{bmatrix} 0.002 & 0.0001 \\ 0.08 & 0.00001 \end{bmatrix}$. According to Eq. (2), we have $w' = sign(w)(|w|^{\beta} + \alpha) = \begin{bmatrix} 0.002 & 0.0001 \\ 0.08 & 0.00001 \end{bmatrix} \neq \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$. In such a scenario, w' can still performs the convolution operation. Therefore, the bias tensor, α, can help the generated kernels extract the salient features showed in Step 2 of Fig. 2.

With this in place, w' extracts input features as the standard kernels in the forward propagation. Assuming there is input image x and outputs m feature maps after performing convolution operation and passing through a or several non-convolution activation functions (e.g., $Batch\_Norm + ReLU$) in a module, such a process can be shown as follows.

$$y = \sum_{i=1}^{m} g(w' \times x) \tag{3}$$

where $g(\cdot)$ indicates the non-convolution activation functions. We forward perform Eq. (3) in each layer until all layers extract the corresponding data features.

After the forward propagation, we utilize cross entropy [46] to update the parameters of each client's network. Different from the traditional FL method, we only update the parameters of base kernels and then discard the generated kernels during back propagation by viewing the generated kernels as local variables and regarding the base kernels as instance variables in **Fed-KGF**, because the local variables are automatically released. Here, we call the kernels that the parameters are updated as representative kernels, which we target to transmit as shown in Step 3 of Fig. 2.

Although the term Kernel Generation Function is a commonly used concept and appeared in other studies as well, we have differences. In our paper, the kernel generation function indicates that we generate kernels to render an incomplete network to be a complete one for taking in inputs and then extracting their features, based on a few base kernels, with $w' = sign(w)(|w|^{\beta} + \alpha)$ where w denotes the base kernels and both β and α indicate the tensor and bias, respectively. However, in other studies (e.g., [20] [21]), the "kernel generation function" is to generate the involution kernels (i.e., inverting the convolution) from a pixel point of the input feature tensor, with $H_{i,j} = \phi(X_{i,j}) = W_1\sigma(W_0X_{i,j})$ where $X_{i,j}$ is a pixel point in row i and column j of input feature tensor. Another example of "kernel generation function" study is to generate convolution kernels [22]. Yet, this method is similar to studies [20] [21], and it also utilizes input data to generate kernels rather than a few base kernels, with $X = \sigma((\alpha_1 \cdot W_1 + \ldots + \alpha_n \cdot W_n) * x)$ where $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ denote the scalar weights learned through gradient descent and x indicates the input data. Obviously, we have different mathematical expressions, kernel generation strategies, and computation operations.

3.3 Design of Module Uploading Method

Since each client holds different data samples, the training of each local model could give rise to the client-variance. This indicates that the local models have difference to each other after downloading the parameters of the global model to each client with averaging method [26]. With the epoch increasing, the value of client-variance becomes larger, causing the issue of client-drift in the global model and thus resulting in the low performance for this global model [26]. Here, we innovatively present the permutation module uploading method to address this issue.

Usually, a CNN is assembled by multiple modules, and a module is stacked by multiple functions. For example, one convolution function + several non-convolution functions, which corresponds to $Conv + Batch\_Norm + ReLu/Pooling$ functions. In the traditional FL, the total transmitted parameters of a local network can be expressed as $P_{CNN} = \sum_{i=1}^{k} m_i$ where P_{CNN} represents the total

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

Algorithm 1: Training procedure of **Fed-KGF**.

Input: Original data x, Original label y_{true}
Output: Prediction results from the global network.

n clients, and an incomplete network has base kernels, w with parameters θ. SGD optimizer and $Cross\ Entropy$ loss function and index i (e.g., w_i, α_i, and β_i).

for *each client i∈ [1, n]* **do**
  Constructing an incomplete CNN with base kernels, w;
end

for *each epoch t=1, 2,...* **do**
  Client-side Procedure:
  for *each client i ∈ [1, n]* **do**
    $w' \longleftarrow$ *Generating_Kernels(w)*;
    $y_{pred} \longleftarrow$ forward(x);
    Updating the parameters of w' by descending its stochastic gradient;
    $\bigtriangledown_\theta \sum_1^m \{-y_{true}\ log(y_{pred})\}$, m indicates the batch_size;
    discarding the generated kernels, w';
  end
  Modules transmission; ▷ *Uploading one or few modules from client networks in a permutation way to the server.*
  Server-side Procedure:
  Modules Aggregation and distributing global model to all clients; ▷ *Receiving uploaded modules from all clients and then combining them to form the global model. After that, distributing it to each client.*
end
Function *: Generating_Kernels(w)*:
  Initializing w' by "coping" the first dimension of w;
  $\alpha \longleftarrow$ Generating a tensor from the range of [0.00001, 0.1];
  $\beta \longleftarrow$ Generating a tensor from the range of [2, 10];
  for $w_i \in w,\ \alpha_i \in \alpha,\ \ \beta_i \in \beta$ **do**
    $w_i^{'} \longleftarrow sign(w_i)(|w_i|^{\beta_i} + \alpha_i)$; ▷ This process indicates the Eq. (2)
  end
  return w';
End Function

parameters of a network. k indicates the number of modules and m_i means a certain module in this network. Obviously, the performance of the global model is affected by the averaged parameters, because averaging the averages breaks the fundamental rules of math such that some studies insist that averaging method is derived by the "wrong math" [47] [48]. In our idea, we consider each module as equally important. Since each client holds one network and these networks have the same architecture, we innovatively upload one or few modules (i.e., $\sum_{i=1}^{k^{'}} m_i$ where $1 \leq k^{'} << k$) from a client in a permutation way, which is shown in Step 4 of Fig. 2. Specifically, the upload process is described as follows:

1) We count the number of clients, and then generate a set of random integer values. The number of this set is the same as that of clients.
2) We select one or few modules from clients, according to the numbers of clients and modules. Assuming there are 3 clients and a network holds 3 modules, and a set of random values is [2,1,3], we select $2nd$ module, $1st$ module, and $3rd$ module from client 1, client 2, and client 3. If there are 3 clients and a network holds 5 modules, we split 5 modules into 3 groups (e.g., taking $1st$ and $2nd$ modules as the first group, and viewing $3rd$ and $4th$ modules as the second group). We select the three groups as the case of [2,1,3].
3) We upload all selected modules to the server. It aggregates those modules to establish a global network, and then distributes its parameters to each local network.

Obviously, the transmitted parameters are further reduced, because we upload a module or few modules to server. Moreover, this can reduce the issue of client-drift. Defining θ as the parameters of a certain local model (i.e., $f(x;\theta)$), and $\overline{\theta}$ as the averaged parameters of the global model, a updated local model is $\triangle\theta = \theta - \overline{\theta}$. With this, we have the client-variance: $\delta^2 = \frac{1}{n}\sum_{i=1}^{n} ||\triangle\theta_i||^2$. The larger δ^2 is, the more serious the client-drift is, given that the variance is used to measure the degree of drift. This demonstrates the influence of the averaging method on the performance of the final global model, because the larger the client-drift is, the worse performance the global model holds [26]. In our **Fed-KGF**, the global model is assembled by the uploaded modules and it discards the averaging. Given that the calculation of averaging is in the dimension manner, we compare averaging with the our idea from the dimension aspect. For example, assuming there are three local models, named $\mathcal{A}$: $[0.00, -0.17, -0.05]$ and $\mathcal{B}$: $[0.01, -0.10, 0.06]$ and $\mathcal{C}$: $[-0.04, -0.09, -0.01]$ in which each entry represents a certain module (e.g., $\mathcal{A}_i$ indicates ith module in local model $\mathcal{A}$). We have the modules (θ_F : $[0.00, -0.10, -0.01]$) within the global model for the **Fed-KGF** and $\overline{\theta}$: $[-0.01, -0.12, 0.00]$ for the averaging, and then calculate $\triangle\theta$ in the dimension way. With this, we have $||\mathcal{B}_1 - \overline{\theta}_1||^2 = ||0.01 - (-0.01)||^2$=0.0004 and $||\mathcal{B}_1 - \theta_{F_1}||^2$=0.0001 (i.e., 0.0004>0.0001), $||\mathcal{A}_1 - \overline{\theta}_1||^2$=0.0001 and $||\mathcal{A}_1 - \theta_{F_1}||^2$=0.00 (i.e., 0.0001>0.0000), $||\mathcal{C}_1 - \overline{\theta}_1||^2$=0.0009 and $||\mathcal{C}_1 - \theta_{F_1}||^2$=0.0016 (i.e., 0.0009<0.0016). Obviously, the

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

performance of $1st$ updated module in **Fed-KGF** is better than averaging, because the values of $\triangle\theta$ for our method is less than that for averaging in most scenarios. Other modules have the similar performance as the $1st$ module. With such a place, the value of $\triangle\theta$ in **Fed-KGF** is smaller than that in averaging, thus obtaining smaller variance. Hence, **Fed-KGF** reduces the issue client-drift, thus improving the performance of the global model.

Another advantage of our strategy is to obtain better optimization than averaging. Let $f_{global_{AVG}}(x;\theta) = \frac{1}{n}f(x;\theta)$ in the traditional FL where $f_{global}(x;\theta)$ indicates the parameters of the global model and $f(\cdot)(x;\theta)$ indicates the parameters of a local model and there is n clients, and let $f_{global_{FED-KGF}}(x;\theta) = m_k \circ m_{k-1} \cdots m_1$ in the **Fed-KGF** where k indicates the number of modules in a model, it is obviously observed that the parameters of the global model in **Fed-KGF** are unchanged as the local models. Note that there is $f_{local}(x;\theta) = f_{global}(x;\theta)$ operation in the client-side after the server downloads its model to each client for FL. This could result in that a local model is far away from its optimal status when this model approximates the optimal status after training, because its parameters is averaged after performing $f_{local}(x;\theta) = f_{global}(x;\theta)$. Although the averaging method can reach to the optimal status, it needs more training cost (e.g., more epochs). However, **Fed-KGF** does not adopt averaging, but solely assembles all uploaded modules. Therefore, the optimization can be improved. The pseudo-codes of **Fed-KGF** is presented in **Algorithm 1**, and we state its convergence as below.

Proposition 3. Considering the global model aggregates all k modules in a permutation way, and let a module within in the global model be m_i and $i \in [1, k]$. If local models get convergence, then the global model holds the convergence.

Proof. The architecture of the global model is the same as that of the local model in FL. Let a local model be $f_i(x;\theta_i) = m_{i,k}(\theta_{i,k}) \circ m_{i,k-1}(\theta_{i,k-1}) \circ \cdots m_{i,1}(\theta_{i,1})$ where i denotes ith local model and a network has k modules. Let the global model be $f_{global}(x;\theta_{global}) = m_k(\theta_k) \circ m_{k-1}(\theta_{k-1}) \circ \cdots m_1(\theta_1)$ where m modules are from different local model. Since FL performs $f_i = f_{global}$ operation after distributing the parameters of the global model to each client, and local models certainly get converged after training, f_{global} in our method also holds convergence. If not so, assuming local models get converged and the global model does not hold the convergence and f^* be the optimal status, the distance between current status of this model and optimal status of the same model is $\|f_i - f^*\| \to 0$ after training [28] [1]. When the global model does not reach the optimal status, the local model is also far away from the optimal status due to $f_i = f_{global}$ operation, given that the modules from converged local models are uploaded to the server and the global model does not update any parameters. This obviously contradicts the assumption. Therefore, f_{global} holds the convergence.

Note that **Algorithm 1** performs the classification, so we utilize the *Cross Entropy* loss to update the parameters of networks. One can choose other loss functions to carry out different tasks (e.g., mean-square error loss or binary cross entropy for object detection).

3.4 Analysis of FED-KGF

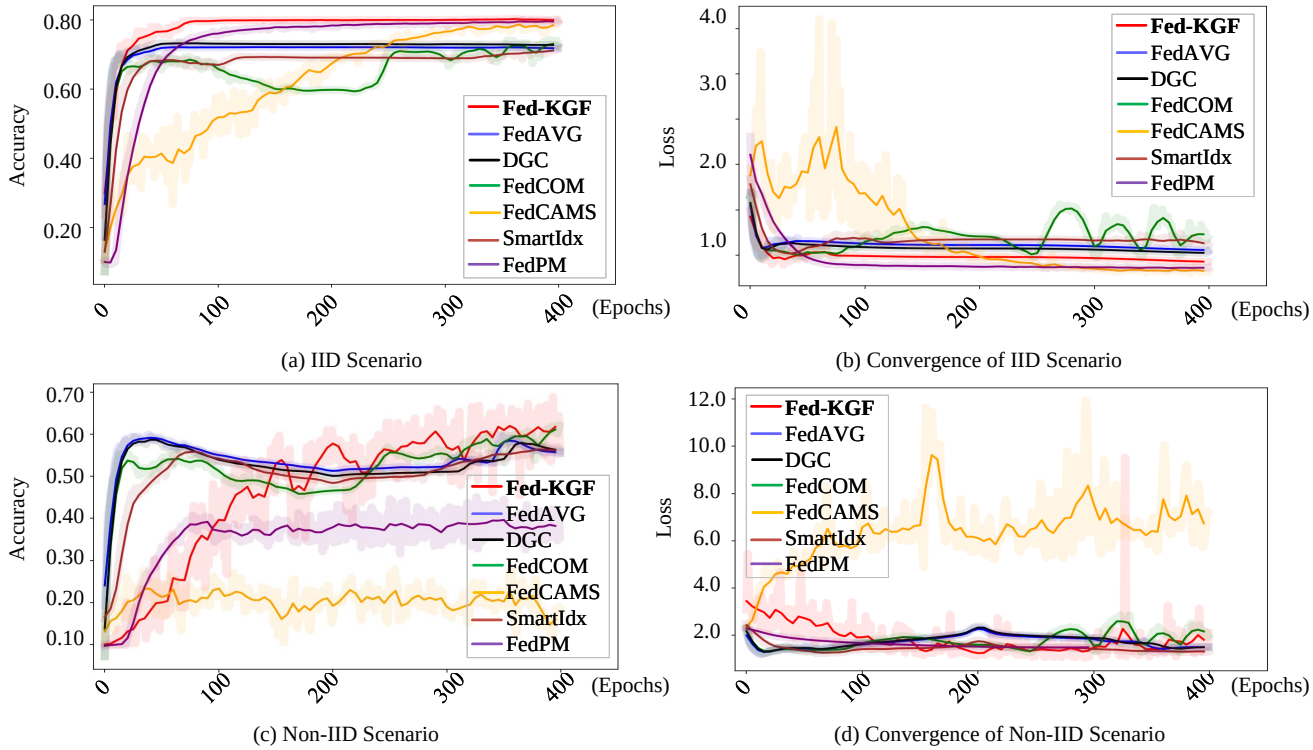
In this sub-section, we provide an in-depth analysis of **FED-KGF** by comparing it with other well-known significant FL variants. Assuming the size of a kernel is $k \times k$, the number of channels is c and the number of kernels is n in a standard convolution layer, the number of learnable parameters can be expressed by $c \times k \times k \times n$. As to **FED-KGF**, the number of learnable parameters is $c \times k \times k \times m$ with $m << n$, which means that the number of learnable parameters of **FED-KGF** is significantly smaller than the standard CNN. Note that since non-convolution functions in a certain module take few, even zero, parameters and they exist both **FED-KGF** and the standard CNN, we focus on the convolution layer to evaluate the parameter reducing ratio, and it is mathmatically expressed as: $\frac{P_{CNN}}{P_{\textbf{FED-KGF}}} = \frac{c \times k \times k \times n}{c \times k \times k \times m}$. For convenient explanation, we evaluate the parameter reducing ratio in a certain convolution layer, so m indicates the number of representative kernels in this layer. Take a convolution layer of Resnet18 with 128 kernels as the example. We set the number of representative kernels to 16. In this way, we obtain the parameter reducing ratio in this layer and its value is 8 $\frac{128}{16} = 8$. Nevertheless, Resnet18 includes 4 convolution layers with 128 kernels. Besides, there are additional 4 layers with 256 kernels, 4 layers with 512 kernels, and 5 layers with 64 kernels in this network. Hence, the total parameter reducing ratio is $\frac{c \times k \times k \times 3904}{c \times k \times k \times 16} \approx 244$ where $3904 = 128 \times 4 + 256 \times 4 + 512 \times 4 + 64 \times 5$. Note that we could upload more than 1 module to the server from a network due to the number of clients, the parameter reducing ratio of **FED-KGF** still outperforms other FL variants. More detailed results are shown in experiments.

The parameters of SmartIdx [42] are determined by the kernels and the number of index, with $C = C_{ip} + C_{kp} + C_{pp}$ where C_{ip} indicates the individual package (i.e., ip) and it is composed by $W_{ip} + G(i_{ip})$; C_{kp} denotes the kernel package (i.e., kp) and it is composed by $\sum_{i=1}^{k} S_i n_i + G(i_{kp})$; C_{pp} indicates the pattern package (i.e., pp) and it is composed by $\sum_{j=0}^{p} S_j + G(i_{pp})$. In the three terms, $G(\cdot)$ indicates the cost of *Golomb* encoding, and W_{ip} means the number of parameters in ip. After sparsification with top-k (here we set $k = \frac{1}{32}$ as the original work) and *Golomb* encoding, the parameter reducing ratio can be expressed: $W_{ip} + \sum_{i=1}^{k} S_i n_i + \sum_{j=0}^{p} S_j$. Still take Resnet18 as the example, the total parameter reducing ratio is 25.6.

Since DGC [37] propagates the larger gradients while stores the smaller gradients to accumulate and update for later propagation, its parameters heavily relies on the top-k. After sparsification with top-k, the number of parameters has reduced to $\frac{1}{32}$ of the original one when we set $k = \frac{1}{32}$. Moreover, each parameter needs an index with 24bits. The total parameter reducing ratio is 18.28. FedCAMS [17] utilizes *Scaled sign* method to quantize parameters, with $\sum_{i=0}^{|x|} d(sign(x_i))$ where $|x|$ indicates the number of parameters of a model, x. $d(\cdot)$ measures the bits of type of data, and x_i denotes i-th parameter within x. The total parameter reducing ratio is 32. FedCOM [14] requires the extra bandwidth to track the information of uploaded gradients and the size of extra bandwidth is the same as the that of uploaded gradients, so the reduced parameters are not significant, and its total parameter reducing ratio is 2.0.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,



(a) IID Scenario (b) Convergence of IID Scenario

(c) Non-IID Scenario (d) Convergence of Non-IID Scenario

Fig. 3: The classification performance on CIFAR10.

4 Experiments

4.1 Experimental Setting

For the experiments, three public datasets are studied, which are CIFAR10, CIFAR100, and COCO2017 datasets. The first two datasets are used to validate the performance of classification, and both of them have 50000 training images and 10000 test images from 10/100 categories (each category has 5000 images in training set). We utilize **top-1 accuracy** (the higher score, the better) as the evaluation metric. COCO2017 is utilized to validate the performance of detection, and it includes 118287 training images and 40670 test images, with 80 object categories. The **mean average precision (mAP)** and its variants (e.g., AP50, AP75, APM, and APL) (the higher scores, the better) are adopted as the evaluation metrics for object detection.

Six SOTA FL methods are employed as baselines, which are **FedAvg** [1], **DGC** [37], **FedCOM** [14], **FedCAMS** [17], **SmartIdx** [42], and FedPM [43]. These models are representative on reducing parameters for fast transmission between clients and server, and their hyper-parameters are identical to original works for fair comparison. We compare our **Fed-KGF** with baseline models on those datasets in both IID and non-IID scenarios. For each scenario, the number of base kernels is set to 16, and all models run 400 epochs. The GPU is Nvidia 3080Ti. As to experimental settings, we employ 10 clients and 1 server for all cases. One model is deployed on each node. For example, each node holds a Resnet18 network for classification on CIFAR10/100 datasets or a YOLOX-S network for object detection on COCO2017 dataset.

Note that the number of modules within both Resnet18 and YOLOX-S is larger than 10 and we only have 10 clients. Therefore, we group 18 modules (take Resnet18 as the example) into 10 groups. In other words, some groups include 2 modules, and they are randomly selected in a permutation way at each epoch. Moreover, we transmit one group from the client into the server at each epoch, and selecting such a group still relies on the permutation strategy in our **Fed-KGF**. The case of YOLOX-S is similar to Resnet18.

4.2 CIFAR10 for Classification

We first validate all models on the CIFAR10 dataset. In this case, the training samples are divided into 10 sub-sets, and each sub-set has the same number of images and all 10 categories for IID scenario. One client solely holds one sub-set. For non-IID scenario, the training samples are also divided into 10 sub-sets, with the same quantity. However, each sub-set has one main category, which takes 80% proportion of samples in this sub-set. For example, sub-set 1 contains 4000 car images (i.e, $\frac{4000}{5000}$), and sub-set 2 has 4000 horse images. The rest of each sub-set holds other 9 categories. The experimental results are shown in Fig. 3. Obviously, **Fed-KGF** outperforms other FL models in both scenarios (See Fig. 3 (a) and (c)). Only the accuracy from **Fed-KGF** is over 80% (i.e., 81%), while others are worse than 80% (FedPM achieves 79.8% accuracy). Fig. 3 (b) and (d) illustrate that **Fed-KGF** can get convergence. Models like FedCOM and FedCAMS are difficult to get convergence.

We also compare the transmitted parameters and the consumed times as listed in TABLE 1. Since both scenarios have the same experimental settings (e.g., the same architectures of Resnet18 and GPU as well as the training epochs), they have similar transmitted costs. Hence, we only report one scenario as shown in TABLE 1. In TABLE 1, each entry

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

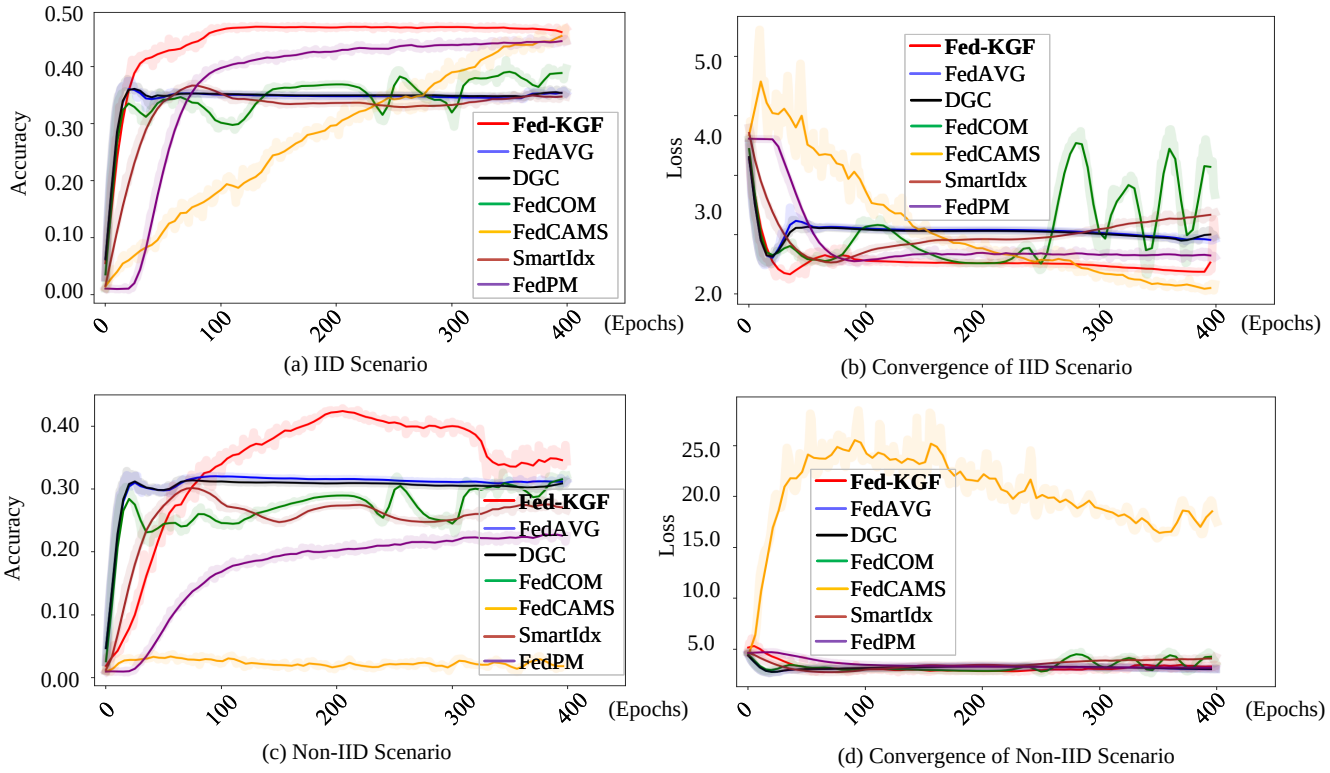


Fig. 4: The classification performance on CIFAR100.

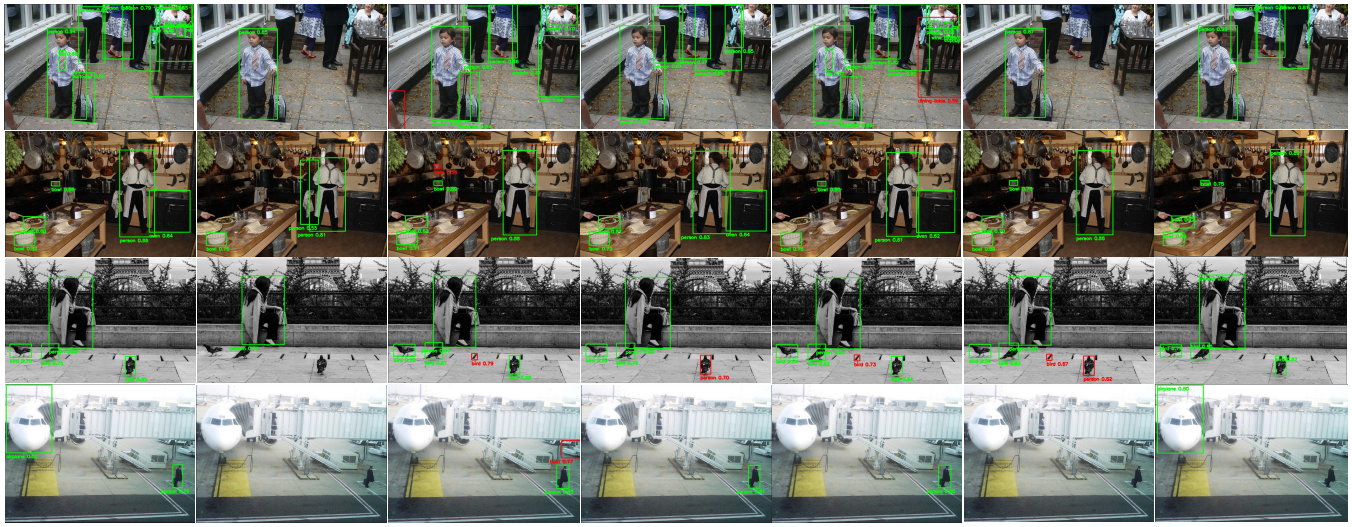


Fig. 5: The object detection performance of all models on COCO2017 in IID scenario.

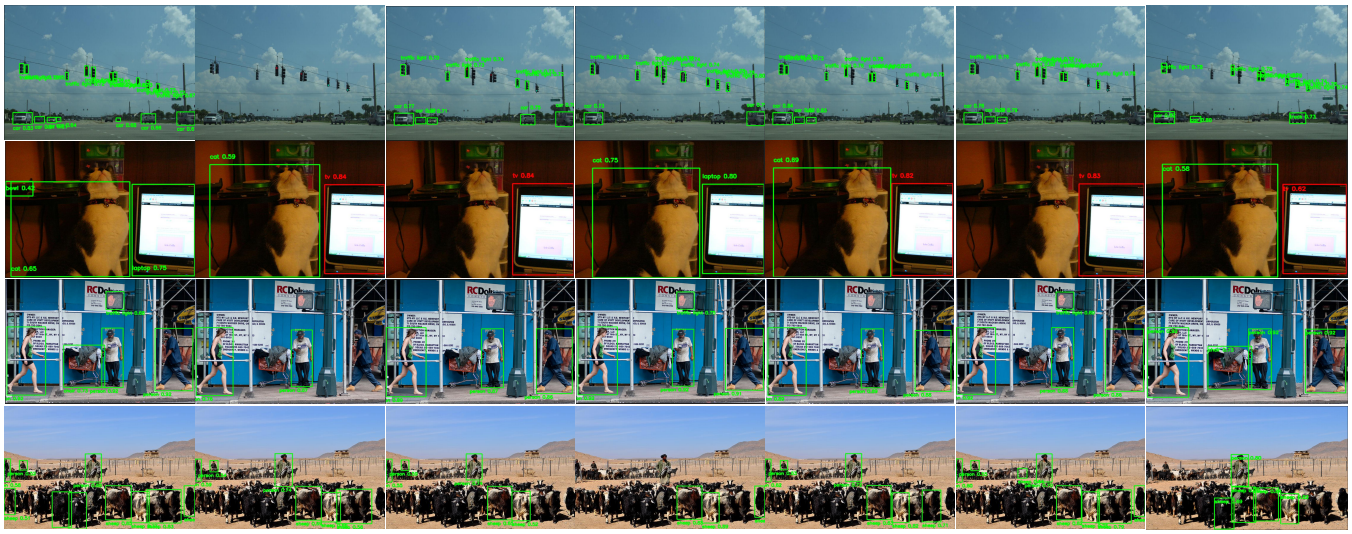
represents the quantity of transmitted parameters of per epoch. For example, 10 clients totally transmit 8.46MB parameters to the server in one epoch, which spends 0.774ms in **Fed-KGF**. Here, we view the transmitted cost of FedAVG as the benchmark. $\top_R(\downarrow)$ denotes the reduction rate of parameters of models (the smaller, the better), with $\frac{PM-BM}{BM}$ where PM denotes the number of parameters of current model and BM means that of the benchmark. The experimental results show that **Fed-KGF** achieves the lowest transmission cost (i.e., the minimum transmitted time and parameters) among all models, where the number of transmitted parameters are reduced by 32.97% when comparing to FedCAMS. Besides, we prove the necessity of module uploading by removing it, and the results are shown at the row of *Fed-KGF without MU* in TABLE 1. Obviously, the cost arises after removing module uploading, which validates its necessity. The results demonstrate the effectiveness of **Fed-KGF** over SOTA FL models in significantly reducing parameters for faster transmission.

4.3 CIFAR100 for Classification

We then validate all FL models on CIFAR100 dataset. In this case, we also consider two scenarios: IID and non-IID.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,



(a) Fed-KGF (b) FedAvg (c) DGC (d) FedCOM (e) FedCAMS (f) SmartIdx (g) FedPM

Fig. 6: The object detection performance of all models on COCO2017 in non-IID scenario.

TABLE 1: The transmission costs of all models on CIFAR10, and we view the transmission cost of Fed-AVG as the benchmark. The ablation study (i.e., only removing the module uploading method) is named as Fed-KGF without MU.

Model	Time (ms)	Parameters (MB)	$\top_R(\downarrow)$
Fed-KGF	**0.774**	**8.46**	**-97.08%**
Fed-KGF without MU	0.838	13.20	-95.44%
FedCAMS	1.289	13.32	-95.40%
SmartIdx	0.892	20.47	-92.92%
DGC	1.032	36.16	-87.50%
FedPM	0.912	91.55	-68.36%
FedCOM	1.140	144.65	-60.37%
Fed-Avg	1.189	289.31	0.00%

For the first one, images from CIFAR100 are divided into 10 sub-sets, and each sub-set has 100 categories, with the same quantity as the CIFAR 10 case; for the other scenario, images are still divided into 10 sub-sets, but each sub-set only has 10 main categories. Different from the CIFAR10 non-IID scenario, each category takes 8% proportion of images, so 10 main categories take 80% proportion of images in this sub-set. For example, sub-set 1 has beaver, dolphin, otter, seal, whale, aquarium fish, flatfish, ray, shark, and trout categories, and each category has 400 images. The rest is from the other 90 categories. Each client holds one sub-set and deploys one Resnet18. The experimental results are shown in Fig. 4, which shows that **Fed-KGF** achieves the best classification performance over baseline models. In IID scenario, our **Fed-KGF** obtains 46.9% accuracy, which is the highest among all models, given that FedPM and FedCAMS achieve 44.2% and 44.6%, respectively. In the non-IID scenario, our **Fed-KGF** still holds the highest accuracy among all models. Besides, we compare the transmitted costs of baselines with **Fed-KGF**. Since CIFAR100 has the same number of training/test images and the deployed model is Resnet18 as the CIFAR10 case, the transmission cost is similar to the results reported in TABLE 1.

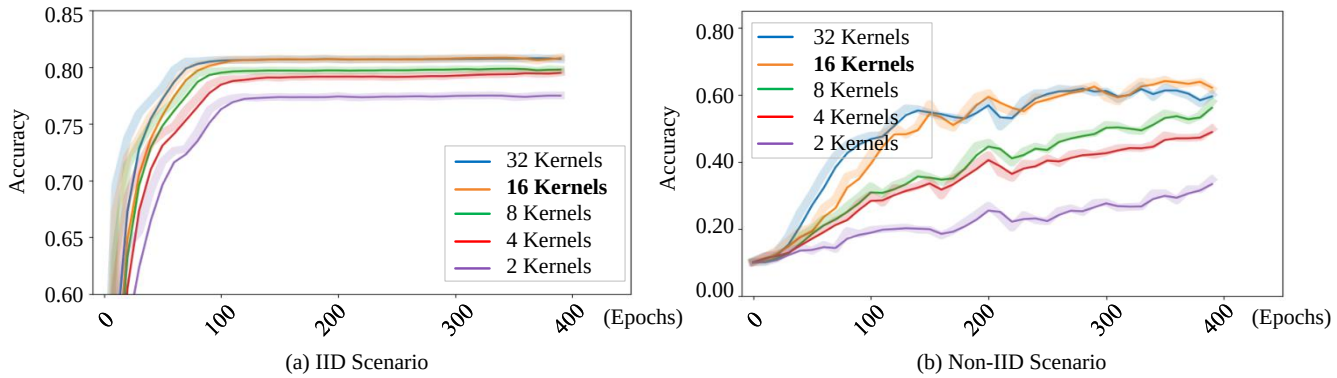
4.4 COCO2017 for Object Detection

Given the large amounts of training images in COCO2017 dataset and the limited GPU memory, we take YOLOX-S [11] (8.94MB parameters) as the detector, and deploy 10 YOLOX-S models (89.4MB parameters) to 10 clients and a YOLOX-S to the server. In IID scenario, the training images are also divided into 10 sub-sets, and each sub-set has all categories and holds the same number of images. As to non-IID scenario, all training images are still divided into 10 sub-sets. However, each sub-set has 8 different main categories, and they take 80% proportion of images in each sub-set as CIFAR10/100 cases. The rest 20% images are from other categories. The detected results of both IID and non-IID scenarios are shown in Fig. 5 and Fig. 6, respectively. In Fig. 5, it can be observed that FedAvg (e.g., [2, 2]), DGC (e.g., [2, 3] and [4, 3]), FedCOM (e.g., [2, 4] and [4, 4]), FedCAMS (e.g., [4, 5]), SmartIdx (e.g., [1, 6]), and FedPM (e.g., [1, 7] and [4, 7]) miss to detect some objects, where the first number indicates the row and the second number indicates the column in [,]. Furthermore, FedAvg (e.g., [4, 2]) and SmartIdx (e.g., [4,6]) even fail to detect any objects in some scenarios. DGC (e.g., [2-4, 3]), FedCOM (e.g., [3, 4]), FedCAMS (e.g., [1, 5] and [3, 5]), and SmartIdx (e.g., [3, 6]) wrongly identify a car as a boat or view a bird as a person. **Fed-KGF** is capable of detecting more objects and holding the less identification errors than all baseline models. The performance of all models in non-IID scenario is shown in Fig. 6. From Fig. 6, it is clearly to observe that FedAvg (e.g., [2, 2]), DGC (e.g., [2, 3]), FedCAMS (e.g., [2, 5]), SmartIdx (e.g., [2, 6]), and FedPM (e.g., [2, 7]) wrongly identify laptop as the TV. FedCOM (e.g., [3-4, 4]) and FedPM (e.g., [1, 7]) miss to detect objects. Only **Fed-KGF** is capable of detecting more objects with less identification errors than baselines.

We also quantitatively measure the performance of all models, and the quantitative results are reported in TABLE 2 and TABLE 3, respectively. In both tables, AP50 (AP75) indicates that the value of Intersection over Union (IoU) $\geq 50\%$ (IoU $\geq 75\%$ for AP75). APM indicates AP of objects with middle sizes (i.e., from 32 $\times$ 32 to 96$\times$96), while

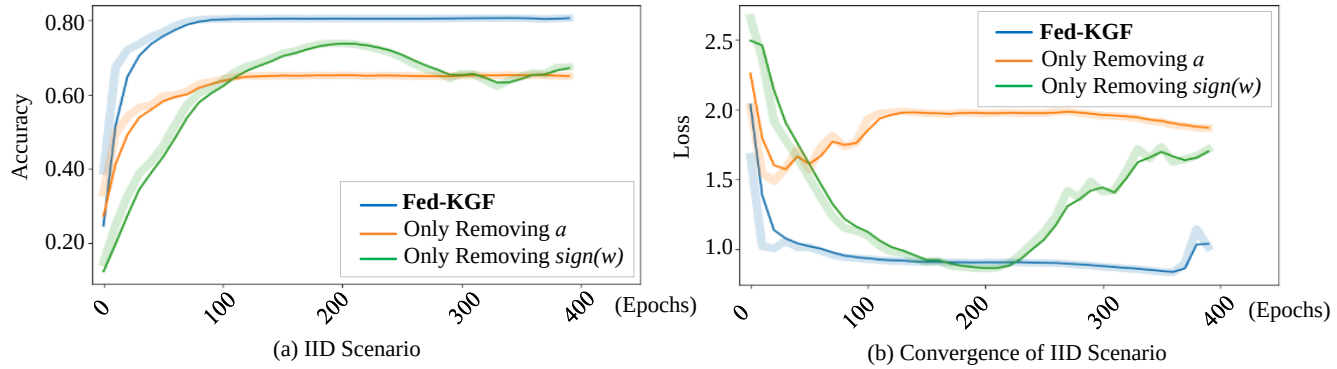
Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,



(a) IID Scenario (b) Non-IID Scenario

Fig. 7: The performance of different numbers of base kernels on CIFAR10.



(a) IID Scenario (b) Convergence of IID Scenario

Fig. 8: The ablation studies on CIFAR10.

APL indicates AP of objects with large sizes (i.e., object size $\geq$ 96×96). Both tables show that **Fed-KGF** has better detection performance than SOTA FL models. Moreover, we have compared the transmission costs of all baseline models with **Fed-KGF**, and the experimental results are reported in TABLE 4. It is noted that both IID and non-IID scenarios have the same experimental settings on the same dataset, their transmission costs are similar to each other, so we only report one of them as shown in TABLE 4. According to the results, we can observe that **Fed-KGF** achieves the lowest cost among all SOTA FL models. Compared to the recent SmartIdx and the FedPM models, the numbers of transmitted parameters of **Fed-KGF** have reduced by 36.49% and 44.64%, respectively. The experimental results further demonstrate the effectiveness and high-performance of **Fed-KGF** in reducing parameters for faster transmission.

TABLE 2: The quantitative results from all FL models on COCO2017 in IID scenario.

Model	mAP	AP50	AP75	APM	APL
Fed-KGF	**0.384**	**0.565**	**0.417**	**0.428**	**0.506**
Fed-Avg	0.363	0.518	0.392	0.410	0.462
DGC	0.381	0.552	0.399	0.425	0.485
FedCOM	0.318	0.493	0.338	0.374	0.388
FedCAMS	0.353	0.540	0.374	0.405	0.450
SmartIdx	0.355	0.539	0.380	0.401	0.451
FedPM	0.313	0.497	0.343	0.362	0.391

4.5 Ablation Studies

Since our kernel generation function includes three parts: $sign(\cdot)$ operation (i.e., $sign(w)$), generated kernels (i.e., $|w^{\beta}|$), and bias tensor (i.e., α), we conduct a series of ablation studies to show the necessity of each component by only removing $sign(w)$ and only removing α from Eq. (2). Considering removing $|w^{\beta}|$ will result in that we cannot take in input data due to the incomplete network, we do not show its performance. We validate the necessity of each component of Eq. (2) on CIFAR10 in the IID scenario, with Resnet18 model. The experimental results of the classification task

TABLE 3: The quantitative results in non-IID scenario.

Model	mAP	AP50	AP75	APM	APL
Fed-KGF	**0.358**	**0.543**	**0.384**	0.404	**0.457**
Fed-Avg	0.346	0.537	0.367	0.392	0.433
DGC	0.325	0.504	0.346	0.372	0.425
FedCOM	0.036	0.058	0.038	0.046	0.388
FedCAMS	0.355	0.540	0.378	**0.407**	0.444
SmartIdx	0.347	0.533	0.370	0.399	0.441
FedPM	0.301	0.479	0.316	0.294	0.376

TABLE 4: The transmission costs on COCO2017 dataset, and Fed-AVG is viewed as the benchmark. The ablation study (i.e., only removing the module uploading method) is named as Fed-KGF without MU.

Model	Time (ms)	Parameters (MB)	$\top_R(\downarrow)$
Fed-KGF	**0.554**	**21.69**	**-93.68%**
Fed-KGF without MU	0.635	30.14	-91.21%
SmartIdx	0.571	32.36	-90.57%
FedPM	0.611	39.18	-88.58%
DGC	0.825	42.87	-87.50%
FedCAMS	0.674	43.08	-87.44%
FedCOM	0.583	171.50	-50.00%
Fed-Avg	0.872	342.99	0.00%

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

are shown in the sub-figure (a) of Fig. 8. It is observed that the classification performance significantly decreases when removing $sign(w)$ or α. Sub-figure (b) shows the corresponding performance of convergence. Clearly, the model without $sign(w)$ or α is difficult to get convergence. In other words, removing any component of Eq. (2) will result in the performance downgrading. However, we can get the desired results as shown in the curves from **Fed-KGF** in the two sub-figures if we reserve those components. As to the module uploading strategy, its necessity has been proved at the row of *Fed-KGF without MU* in TABLEs 1 and 4.

4.6 Discussion

Note that the number of base kernels is empirically evaluated on several tests. We set the number of base kernels to 2^1 (2), 2^2 (4), 2^3 (8), 2^4 (16), 2^5 (32) in both IID and non-IID scenarios on the CIFAR10 dataset, with 10 clients and 1 server, given that the number of parameters in a CNN is usually the power of 2 [28]. It is found from Fig. 7 that the larger number of base kernels outperforms the smaller number of base kernels, and the best performance is achieved with 2^4 (i.e., 16) base kernels. Hence, we set the number of base kernels to 16 for **Fed-KGF** in this paper.

Given that the FL model is usually benchmarked by classification [1] and object detection [25], we also follow the criterion. From the experimental results (See Figs. 3-6, TABLEs 1-4), we can observe that **Fed-KGF** not only holds the least transmitted parameters of models for faster transmission but also achieves the best performance among all SOTA FL models. The recent FL variants, like FedCOM and FedCAMS, are difficult to get convergence. Moreover, we found a new challenge of FedCAMS, which fails to classify categories in non-IID scenario as shown in Fig. 3 (c) and Fig. 4 (c). As to FevAvg, DGC, and SmartIdx models, they show the similar performance in CIFAR10/100 classification cases. In the object detection case, the baseline models basically hold the same issues (e.g., false detection and missed detection), in which FedCOM has the worst detection performance among all models in both IID and non-IID scenarios. As to other baseline models, their detected quantities and categories are worse than **Fed-KGF** as shown in TABLE 2 and TABLE 3.

Since most real-world deep-learning applications (e.g., industrial image analysis [49] [50], face identification [51], and CT detection [52]) focus on classification and detection [28], this paper adopts Resnet18 and YOLOX-S to validate our idea. Moreover, our idea can be viewed as a plug-in to equip other FL variants to improve their performance on reducing the transmitted parameters.

5 Conclusion

In this paper, our approach seeks to significantly reduce parameters for fast transmission in FL. We present Fed-KGF, which generates kernels to take in the inputs in clients and then discards those generated kernels when transmitting their parameters to the server. With this, we solely transmit the representative kernels between clients and the server. The contributions of our **Fed-KGF** are: 1) utilizing "generation mechanism" to reduce parameters for faster transmission; 2) proposing a new module transmission method to further reduce the transmitted parameters; 3) presenting 11% higher classification (3.64% detection performance) and achieving 33% (37%) lower transmitted parameters compared with the SOTA FL variants; and 4) providing new insights into the understanding of FL variants.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (2022YFF1101100).

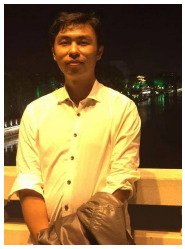
References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.
[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Fl in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
[3] Q. Yang, L. Fan, and H. Yu, *Federated Learning: Privacy and Incentive*. Springer Nature, 2020, vol. 12500.
[4] Y. Gao, L. Zhang, L. Wang, K.-K. R. Choo, and R. Zhang, "Privacy-preserving and reliable decentralized federated learning," *IEEE Transactions on Services Computing*, 2023.
[5] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor, "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3388–3401, 2021.
[6] H. Ko, J. Lee, S. Seo, S. Pack, and V. C. Leung, "Joint client selection and bandwidth allocation algorithm for federated learning," *IEEE Transactions on Mobile Computing*, 2021.
[7] T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, "In-network computation for large-scale federated learning over wireless edge networks," *IEEE Transactions on Mobile Computing*, 2022.
[8] T. Guo, S. Guo, J. Wang, X. Tang, and W. Xu, "Promptfl: Let federated participants cooperatively learn prompts instead of models-federated learning in age of foundation model," *IEEE Transactions on Mobile Computing*, 2023.
[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[11] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.
[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
[13] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 2021–2031.
[14] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *International Conference on Artificial Intelligence and Statistics*, 2021, pp. 2350–2358.
[15] H. Gao, A. Xu, and H. Huang, "On the convergence of communication-efficient local sgd for federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7510–7518.
[16] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "Fetchsgd: Communication-efficient federated learning with sketching," in *International Conference on Machine Learning*, 2020, pp. 8253–8265.
[17] Y. Wang, L. Lin, and J. Chen, "Communication-efficient adaptive federated learning," in *International Conference on Machine Learning*, 2022, pp. 22 802–22 838.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

[18] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.
[19] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations," *Neural Information Processing Systems*, vol. 32, 2019.
[20] D. Li, J. Hu, C. Wang, X. Li, Q. She, L. Zhu, T. Zhang, and Q. Chen, "Involution: Inverting the inherence of convolution for visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 321–12 330.
[21] X. Xiang, R. Abdein, N. Lv, and A. El Saddik, "Invflow: Involution and multi-scale interaction for unsupervised learning of optical flow," *Pattern Recognition*, vol. 145, p. 109918, 2024.
[22] B. Yang, G. Bender, Q. V. Le, and J. Ngiam, "Condconv: Conditionally parameterized convolutions for efficient inference," *Advances in neural information processing systems*, vol. 32, 2019.
[23] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*, 2018, pp. 560–569.
[24] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1580–1589.
[25] C. Ding, Z. Lu, F. Juefei-Xu, V. N. Boddeti, Y. Li, and J. Cao, "Towards transmission-friendly and robust cnn models over cloud and device," *IEEE Transactions on Mobile Computing*, 2022.
[26] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*, 2020, pp. 5132–5143.
[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[29] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Neural Information Processing Systems*, vol. 30, 2017.
[30] X. Dai, X. Yan, K. Zhou, H. Yang, K. K. Ng, J. Cheng, and Y. Fan, "Hyper-sphere quantization: Communication-efficient sgd for federated learning," *arXiv preprint arXiv:1911.04655*, 2019.
[31] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar, "Adaptive quantization of model updates for communication-efficient federated learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2021, pp. 3110–3114.
[32] Y. Youn, B. Kumar, and J. Abernethy, "Accelerated federated optimization with quantization," in *Workshop on Federated Learning: Recent Advances and New Challenges (with NeurIPS)*, 2022.
[33] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Uveqfed: Universal vector quantization for federated learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 500–514, 2020.
[34] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. G. Cui, "Federated learning with quantization constraints," *ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8851–8855, 2020.
[35] Y. Wang, Y. Xu, Q. Shi, and T.-H. Chang, "Quantized federated learning under transmission delay and outage constraints," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 323–341, 2021.
[36] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," *arXiv preprint arXiv:2006.10672*, 2020.
[37] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.
[38] L. P. Barnes, H. A. Inan, B. Isik, and A. Özgür, "rtop-k: A statistical estimation approach to distributed sgd," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 3, pp. 897–907, 2020.
[39] S. Lu, R. Li, W. Liu, C. Guan, and X. Yang, "Top-k sparsification with secure aggregation for privacy-preserving federated learning," *Computers & Security*, vol. 124, p. 102993, 2023.
[40] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "Adacomp: Adaptive residual gradient compression for data-parallel distributed training," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
[41] J. Fang, H. Fu, G. Yang, and C.-J. Hsieh, "Redsync: reducing synchronization bandwidth for distributed deep learning training system," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 30–39, 2019.
[42] D. Wu, X. Zou, S. Zhang, H. Jin, W. Xia, and B. Fang, "Smartidx: Reducing communication cost in federated learning by exploiting the cnns structures," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4254–4262.
[43] B. Isik, F. Pase, D. Gunduz, T. Weissman, and M. Zorzi, "Sparse random networks for communication-efficient federated learning," *The eleventh international conference on learning representations*, 2023.
[44] F. Juefei-Xu, V. Naresh Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 19–28.
[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[46] J. Shore and R. Johnson, "Properties of cross-entropy minimization," *IEEE Transactions on Information Theory*, vol. 27, no. 4, pp. 472–482, 1981.
[47] S.-P. Hu, "Simple mean, weighted mean, or geometric mean?" ISPA/SCEA International Conference, San Diego, CA, 2010.
[48] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, and X. Cui, "Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[49] S. Zhou, D. Van Le, R. Tan, J. Q. Yang, and D. Ho, "Configuration-adaptive wireless visual sensing system with deep reinforcement learning," *IEEE Transactions on Mobile Computing*, 2022.
[50] L. Xie, Z. Chu, Y. Li, T. Gu, Y. Bu, C. Wang, and S. Lu, "Industrial vision: Rectifying millimeter-level edge deviation in industrial internet of things with camera-based edge device," *IEEE Transactions on Mobile Computing*, 2023.
[51] Y. Shen, M. Yang, B. Wei, C. T. Chou, and W. Hu, "Learn to recognise: exploring priors of sparse face recognition on smartphones," *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1705–1717, 2016.
[52] J. Wilson and N. Patwari, "Radio tomographic imaging with wireless networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 621–632, 2010.



Dr. Wei Li (Member, IEEE) was undergraduate from the subject of Information and Computing Science in 2008, and receives his Master's degree in Agricultural Engineering from South China Agricultural University in 2012, and gets his Ph.D. degree in Software Engineering from Wuhan University in 2019. He is an Associate Professor of School of Artificial Intelligence and Computer Science at Jiangnan University. He was the visiting student of University of Massachusetts Boston and had visited the The Hong Kong Polytechnic University as Research Assistant. His research interests include Data Mining, Artificial Intelligence, and Federated Learning.

Mr. Zicheng Shen received the B.S. degree in Applied Chemistry from Jiangnan University in 2022. Currently, he is working toward the M.S. degree with the School of Artificial Intelligence and Computer Science, Jiangnan University. His research interests mainly include software engineering and federated learning.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TMC.2024.3423448,

Dr. Xiulong Liu is currently a professor in College of Intelligence and Computing, Tianjin University, China. Before that, he received the B.E. and Ph.D. degrees from Dalian University of Technology (China) in 2010 and 2016, respectively. He also worked as a visiting researcher in Aizu University, Japan; a postdoctoral fellow in The Hong Kong Polytechnic University, Hong Kong; and a postdoctoral fellow in the School of Computing Science, Simon Fraser University, Canada. His research interests include wireless sensing and communication, indoor localization, and networking, etc. His research papers were published in many prestigious journals and conferences including TON, TMC, TC, TPDS, TCOM, INFOCOM, and ICNP, etc. He received Best Paper Awards from ICA3PP 2014 and IEEE System Journal 2017. He is also the recipient of CCF Outstanding Doctoral Dissertation award 2017.

Dr. Mingfeng Wang (M'16) received his B.Eng. degree in mechanical design and automation and M.Eng. degree in mechanical engineering from Central South University in 2008 and 2012, respectively, and the Ph.D. degree in mechanical engineering from the University of Cassino and South Latium, Italy, in 2016. He is currently a Senior Lecturer in Robotics and Autonomous Systems at Brunel University London. His research interests cover novel design & development of humanoid robots, precision farming robots, continuum robots and hexapod robots, which including robotic technologies in terms of mechanical design, kinematic & dynamic analysis, motion planning, motor control, system design & integration, and fabricating & debugging. He has published more than 30 papers that have been presented at peer-reviewed international conferences or journals.

Dr. Chao Ma is currently an Assistant Professor of School of Cyber Science and Engineering at Wuhan University, P.R.China. His research interests include time series analytics, representation learning, deep learning, explainable AI, and big data analytics. He has published over 30 academic papers in major international journals and conference proceedings. He is now the member of IEEE and the professional member of CCF.

Dr. Chuntao Ding (Member, IEEE) received his Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2021. He is currently a lecturer at Beijing Jiaotong University. He also worked as a research assistant with Hong Kong Polytechnic University and as a visiting scholar at Michigan State University. His research interests include service computing, edge computing, and multi-task learning. He received the Outstanding Ph.D. Thesis award of IEEE Technical Committee on Cloud Computing. He has published more than 20 top conference and journal papers, such as IEEE CVPR, IEEE TPDS, IEEE TMC, etc.

Dr. Jiannong Cao (M'93-SM'05-F'15) received the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in 1986 and 1990, respectively. He is currently the Chair Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include parallel and distributed computing, mobile computing, and big data analytics. Dr. Cao has served as a member of the Editorial Boards of several international journals, a Reviewer for international journals/conference proceedings, and also as an Organizing/Program Committee member for many international conferences.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelinesand- policies/post-publication-policies/