24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

RPN: A Residual Pooling Network for Efficient Federated Learning

Anbu Huang[1] and **Yuanyuan Chen**[2] and **Yang Liu**[3] and **Tianjian Chen**[4] and **Qiang Yang**[5]

Abstract. Federated learning is a distributed machine learning framework which enables different parties to collaboratively train a model while protecting data privacy and security. Due to model complexity, network unreliability and connection in-stability, communication cost has became a major bottleneck for applying federated learning to real-world applications. Current existing strategies are either need to manual setting for hyperparameters, or break up the original process into multiple steps, which make it hard to realize end-to-end implementation. In this paper, we propose a novel compression strategy called Residual Pooling Network (RPN). Our experiments show that RPN not only reduce data transmission effectively, but also achieve almost the same performance as compared to standard federated learning. Our new approach performs as an end-to-end procedure, which should be readily applied to all CNN-based model training scenarios for improvement of communication efficiency, and hence make it easy to deploy in real-world application without much human intervention.

1 INTRODUCTION

In the past decade, Deep Convolutional Neural Networks (DCNN) have shown powerful representation and learning capabilities [10, 18], and achieve unprecedented success in many commercial applications, such as computer vision [17, 14], nature language processing [21, 9, 4], speech recognition [12, 32, 27], etc. Same as traditional machine learning algorithms, the success of deep neural network is partially driven by big data availability. However, in the real world, with the exception of few industries, most fields have only limited data or poor quality data. What's worse, due to industry competition, privacy security, and complicated administrative procedures, It is almost impossible to train centralized machine learning models by integrating the data scattered around the countries and institutions [31].

At the same time, with the increasing awareness of data privacy, the emphasis on data privacy and security has became a worldwide major issue. News about leaks on public data are causing great concerns in public media and governments. In response, countries across the world are strengthening laws in protection of data security and privacy. An example is the General Data Protection Regulation (GDPR) [28] enforced by the European Union on May 25, 2018. Similar acts of privacy and security are being enacted in the US and China.

[1] Webank AI Lab, China, email: stevenhuang@webank.com
[2] Nanyang Technological University, Singapore
[3] Webank AI Lab, China, email: yangliu@webank.com
[4] Webank AI Lab, China, email: tobychen@webank.com
[5] The Hong Kong University of Science and Technology, Hong Kong, email: qyang@cse.ust.hk

To decouple the need for model training from the need for storing large data in the central database, a new machine learning framework called federated learning was proposed [20]. Federated learning provides a promising approach for model training without compromising data privacy and security [5, 20, 16]. Unlike traditional centralized training procedures, federated learning requires each client collaboratively learn a shared model using the training data on the device and keeping the data locally, under federated learning scenario, model parameters (or gradients) are transmitted between the server side and the clients on each round. Due to the frequent exchange of data between the central server and the clients, coupled with network unreliability and connection instability, communication costs have became the main constraints and limitations of federated learning performance.

In this paper, we propose a new compression strategy called Residual Pooling Network (RPN) to address the communication costs problem by parameter selection and approximation. As we will see below, our approach can significantly reduce the quantity of data transmission on one hand, and still able to maintain high-level model performance on the other hand, more importantly, our approach can make compression process as an end-to-end procedure, which make it easy to deploy in real-world application without human intervention.

Contributions: Our main contributions in this paper are as follows:

- we propose a practical and promising approach to improve communication efficiency under federated learning setting. Our approach not only reduces the amount of data being uploaded, but also reduces the amount of data being download.
- we propose a general solution for CNN-based model compression under federated learning framework.
- Unlike parameter-encryption based approach, our algorithm based on parameter approximation and parameter selection, which can keep data security without compromising communication efficiency, and is easy to deploy into large-scale systems.

The rest of this paper is organized as follows: We first review the related works of federated learning and current communication efficiency strategies. Then, we introduce our approach in detail. After that, we present the experimental results, followed by the conclusion and a discussion of future work.

2 Related Work

In this section, we will introduce some backgrounds and related works of our algorithm, including federated learning and communication costs.

2.1 Federated Learning

In the traditional machine learning approach, data collected by different clients (IoT devices, smartphones, etc.) is uploaded and processed centrally in a cloud-based server or data center. However, due to data privacy and data security, sending raw data to the central database is regarded as unsafe, and violate the General Data Protection Regulation (GDPR) [28]. To decouple the need for machine learning from the need for storing large data in the central database, a new machine learning framework called federated learning was proposed, a typical federated learning framework is as shown in Figure 1.

In federated learning scenario, each client update their local model based on local datasets, and then send the updated model's parameters to the server side for aggregation, these steps are repeated in multiple rounds until the learning process converges. since local datasets remain on personal devices during training process, data security can be guaranteed.

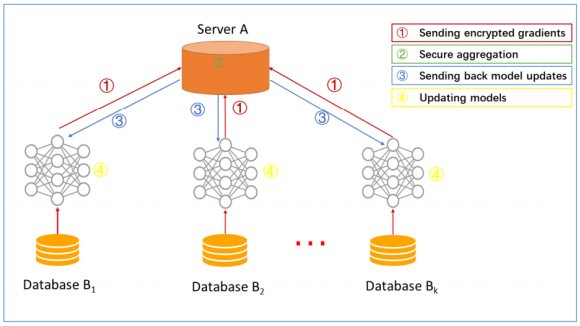

Figure 1: Federated Learning Architecture [31]

In general, the standard federated learning training process including the following three steps [31, 19]:

- ***Local Model Training***: Denote t as the current iteration round. N is the number of clients, G_i^t is the initial model of this round for client i, G_i^t parameterized by w_i^t, that is to say $G_i^t = f(w_i^t)$, D_i is the local datasets of client i. Based on D_i, each client respectively update the local model parameters from w_i^t to w_i^{t+1}, the updated local model parameters w_i^{t+1} are then subsequently sent to the server side.
  To improve communication efficiency, when each round begins, we usually don't require all clients execute local model training, only a subset of M clients should be selected, [20] pointed out that such strategy does not reduce model performance.
- ***Global Aggregation***: The server side aggregates the local model parameters of selected clients, and calculate the global model, we can formalize our aggregation algorithm as follows:

$$w^{t+1} = \sum_{i=1}^{M} f(w_i^{t+1}) \tag{1}$$

  Where f is aggregation operator, some sensible choices of aggregation operator including FedAvg [20], Byzantine Gradient Descent [8], Secure Aggregation [6], Automated Update of Weights.
- ***Update Local Model***: When the aggregation is completed, the server side select a subset of clients again, and send global model G^{t+1} back to the selected clients for next iteration and repeat this cycle until converge.

2.2 Communication Costs

In the standard federated learning training procedure, all model parameters should be exchanged between the server side and the clients, for complicated deep learning model, such as CNN, may comprise millions of parameters, coupled with network unreliability and connection in-stability, making the communication cost a significant bottleneck, as such, how to improve the communication efficiency of federated learning has became an urgent task. The total number of bits that have to be transmitted during model training is given by:

$$S_{total} = \sum_{t=1}^{T} \{F(G^t) + \sum_{i=1}^{M} F(G_i^t)\} \tag{2}$$

where T is the total number of iterations between the server size and the clients, M is the number of clients selected by the server side to update at round t, G^t denotes global model after t times aggregation, $F(G^t)$ is the selective parameter bits download to client side, similarly, $F(G_i^t)$ is the selected parameter bits of client i used to upload to server side. Using equation 2 as a reference, we can classify the current research topics on communication efficiency by the following four aspects:

Iterations frequency: One feasible solution to improve communication efficiency is to reduce the number of communications (see N in equation 2) between the server side and the clients. McMahan et al.[24] proposed an iterative model averaging algorithm called Federated Averaging (FedAvg), and points out that each client can iterate the local SGD update multiple times before the averaging step, thus reducing the total number of communication rounds. The experiments showed a reduction in required communication rounds by 10 to 100$\times$ as compared to FedSGD.

Pruning: Another research topic is to use model compression algorithm to reduce data transmission (see $F(G^t)$ and $F(G_i^t)$ in equation 2), This is a technique commonly used in distributed learning [19, 29]. A naive implementation requires that each client sends full model parameters back to the server in each round, and vice versa. Inspired by deep learning model compression algorithms[11], distributed model compression approaches have been widely studied. Strom el.al[23, 26] proposed an approach in which only gradients with magnitude greater than a given threshold are sent to the server, however, it is difficult to define threshold due to different configurations for different tasks. In a follow-up work, Aji et al.[1] fixed transmission ratio p, and only communicate the fraction p entries with the biggest magnitude of each gradient.

Konecny et.al[16] proposed low rank and subsampling of parameter matrix to reduce data transmission. Caldas et.al [7] extend on the studies in [16] by proposing lossy compression and federated dropout to reduce server-to-participant communication costs.

Importance-based Update: This strategy involves selective communication such that only the important or relevant updates are transmitted in each communication round. The authors in [25] propose the edge Stochastic Gradient Descent (eSGD) algorithm that selects only a small fraction of important gradients to be communicated to the FL server for parameter update during each communication round. the authors in [30] propose the Communication-Mitigated Federated Learning (CMFL) algorithm that uploads only relevant local updates to reduce communication costs while guaranteeing global convergence.

Quantization: quantization is also a very important compression technique. Konecny et.al[16] proposed probabilistic quantization, which reduces each scalar to one bit (maximum value and minimum value). Bernstein et al.[2, 3] proposed signSGD, which quantizes gradient update to its corresponding binary sign, thus reducing the size by a factor of $32\times$. Sattler et.al [22] proposed a new compression technique, called Sparse Ternary Compression (STC), which is suitable for non-iid condition.

Aforementioned approaches are feasible strategies to improve communication efficiency during federated learning training procedure, some of which can reduce the amount of data transferred, while some other can reduce the number of iterations. However, all these solutions are either need to manual setting for hyper-parameter, or break up the original process into multiple steps, as such, cannot implement end-to-end workflow.

3 Methodology

In this section, we formalize our problem definition, and show how to use Residual Pooling Network (RPN) to improve communication efficiency under federated learning framework, our new federated learning framework is shown in Figure 3.

3.1 Symbol Definition

For the sake of consistence in this paper, we will reuse symbol definitions of section 2.1 in the following discussion. Suppose that our federated learning system consists of N clients and one central server S, as illustrated in Figure 1, C_i denotes the ith client, each of which has its own datasets, indicated by $D_i = \{X_i, Y_i\}$ respectively, G^t represents global model after t times aggregation, G^t is parameterized by w^t, that is to say:

$$G^t = f(w^t) \tag{3}$$

For simplicity, and without loss of generality, supposed we have completed $(t-1)$ times aggregation, and currently training on round t, client C_i need to update local model from ${G_i}^{t-1}$ to ${G_i}^t$ based on local dataset D_i. The local model objective function is as follows:

$$\min_{w \in R^d} l(Y_i, f(X_i; w)) \tag{4}$$

The goal of client i in iteration t is to find optimal parameters w that minimize the loss function of equation 4.

let R_i^t represents the residual network, which is given by:

$$R_i^t = G_i^t - G_i^{t-1} \tag{5}$$

Execute spatial mean pooling on R_i^t, it would be changed to ΔR_i^t.

3.2 Residual Pooling Network

The standard federated learning requires exchange all parameters between the server side and the clients, but actually, many literatures [23, 26, 11] had shown that not all parameters are sensitive to model updates in distributed model training. Under this assumption, we propose Residual pooling network, which compress data transmission through parameter approximation and parameter selection, RPN consists of the following two components:

- **Residual Network:** The difference of model parameters before and after the model update is called Residual network. Residual network is used to Capture and evaluate the contribution of different filters, the sum of kernel elements is regarded as the contribution of this filter, only magnitude greater than a given threshold are sent to the server side.
  The purpose of Residual Network is to capture the changes in parameters during training, the author in [13] shown that different filters would have different responses to different image semantics. In light of different distribution of federated clients, each client i would capture different semantic of local dataset D_i, as such, the changes of each layer in local model G_i^t are different, which makes it no need to upload all model parameters.

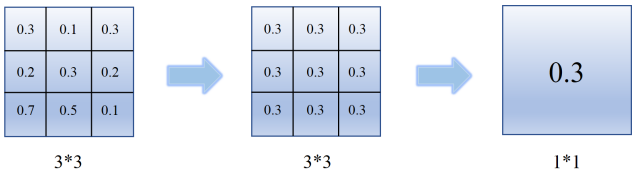

Figure 2: Spatial mean pooling, we calculate the mean value of the kernel, use this value as new kernel with size (1, 1)

- **Spatial pooling:** Spatial pooling is used to further compress model parameters. After we select high sensitive filters through residual network, using spatial mean pooling to compress each filter size from $(size, size)$ to $(1, 1)$, as shown in Figure 2.
  Accord to [15], as the parameters of the previous layers change, the distribution of each layer's inputs changes during model training, we refer to this phenomenon as internal covariate shift, the feasible solution to address the problem is using batch normalization to normalize each layer inputs. Similarly, during federated learning scenario, after local model training, the change in the distribution of network activations due to the change in network parameters during training, spatial pooling of convolutional layer make it normalize each layer by average operation, which makes it reduce the internal covariate shift to some extent.

3.3 Residual Pooling Network - Server perspective

Algorithm 1 RPN: FL_Server

Initialize G^0, and set $\overline{\Delta R^0} = 0$
Send initial model G^0 to each client
for *each round* $t \leftarrow 1, 2, ..., T$ **do**
  $S_t \leftarrow$ *random select* M *clients*
  for *each client* $i \in S_t$ *in parallel* **do**
    $\Delta R_i^t \leftarrow ClientUpdate(i, \quad t, \quad \overline{\Delta R^{t-1}})$
  end for
  RPN aggregation: $\overline{\Delta R^t} = \frac{1}{M} \sum_{i=1}^{M} \Delta R_i^t$
end for

The main function of federated server is used to aggregate the model parameters uploaded from all selective clients, when aggregation completed, the new updated global model parameters are then

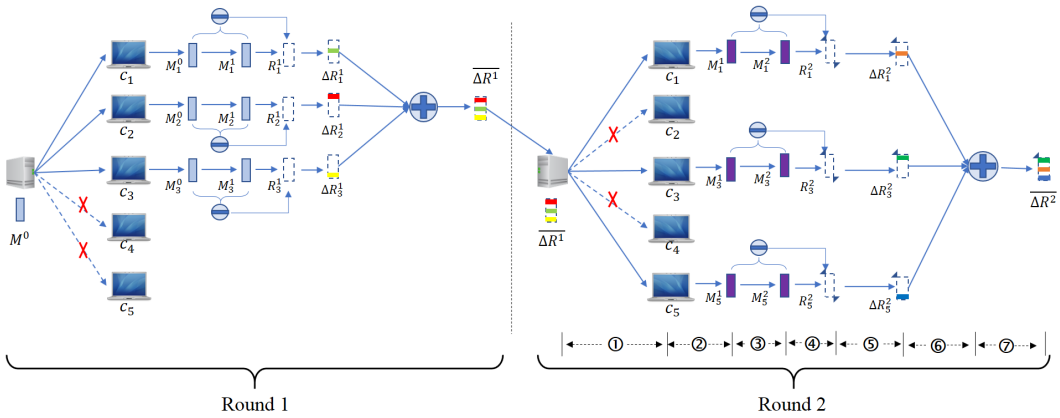

Figure 3: New federated learning workflow of our approach: (1) select clients for local model update. (2) recover local model. (3) local model training based on local datasets. (4) calculate residual network. (5) spatial pooling. (6) send rpn to server side, and do aggregation. (7) send rpn back to selected clients, and repeat this cycle.

subsequently sent back to the clients for next iteration. In this section, we analyze our RPN algorithm from the perspective of federated server, in our new strategy, we don't aggregate raw model parameters, but residual pooling network parameters instead, since only compressed parameters are transmitted, the amount of data downloaded will be significantly reduced.

First, we initialize global model as G^0, and send it to all clients, also set $\overline{\Delta R^0} = 0$. For each round t at server side, also for each client i, we send a triple (i, t and $\overline{\Delta R^{t-1}}$) to all selective clients for local model training, wait until we get response ΔR_i^t from all selective client, the global aggregation can be expressed as follows:

$$\overline{\Delta R^t} = \frac{1}{M}\sum_{i=1}^{M}\Delta R_i^t \tag{6}$$

Repeated this cycle until model converges or maximum number of iterations are satisfied. The entire federated server algorithm can refer to Algorithm 1.

3.4 Residual Pooling Network - Client perspective

Typical federated learning requires each client updates the local model based on local dataset, and then send model parameters to server side for aggregation, however, as previously discussed, not all parameters are sensitive to model updates in distributed model training. In this section, We will analyze RPN algorithm from the perspective of the client, and show how the amount of data uploaded will be significantly reduced.

Without loss of generality, suppose we start at round t. The first step is to recover local model G_i^{t-1}, based on G_i^0 and the sum of $\overline{\Delta R^j}$, where j from 1 to $t-1$, which means:

$$G_i^{t-1} = G_i^0 + \sum_{j=1}^{t-1}\overline{\Delta R^j} \tag{7}$$

After that, based on local datasets D_i, we do local model update, change the model from G_i^{t-1} to G_i^t, and calculate residual network $R_i^t = G_i^t - G_i^{t-1}$, compress R_i^t using spatial mean pooling technique, get the final RPN model ΔR_i^t, send it back to server side for aggregation. The entire federated client algorithm can refer to Algorithm 2.

Algorithm 2 RPN: FL_Client (ClientUpdate)

Input: client id i; round t; $\overline{\Delta R^{t-1}}$

Output: model parameters that sent back to server

Recover model: $G_i^{t-1} = G_i^0 + \sum_{j=1}^{t-1}\overline{\Delta R^j}$

for *each local epoch* $e \leftarrow 1, 2...E$ **do**

  $X \leftarrow$ *random sample dataset with size* B

  $w_i^t \leftarrow w_i^{t-1} - \eta * \nabla l(y_i, f(D_i; w))$

end for

Let $G_i^t = f(w_i^t)$

Calculate residual model $R_i^t = G_i^t - G_i^{t-1}$

Compress R_i^t with spatial pooling, and get ΔR_i^t

Send ΔR_i^t back to server side

3.5 Algorithm Correctness Analysis

Lemma 1. Denote w^{t-1} is global model parameters after $(t-1)$ times aggregation, w_i^{t-1} is the local model parameters of client i before training on round t, and w_i^t is the corresponding parameters after training on round t, under **iid** condition, we should have:

$$\mathbb{E}[\frac{1}{M}\sum_{i=1}^{M}(w_i^t - w_i^{t-1})] = \mathbb{E}[\frac{1}{M}\sum_{i=1}^{M}\Delta R_i^t] \tag{8}$$

In order to validate the correctness of previous equation, we can notice that for each client C_i and local model G_i^t, mean operation on residual model parameters ($R_i^t = w_i^t - w_i^{t-1}$) does not affect expectation, this implies the following equation is satisfied:

$$\mathbb{E}[(w_i^t - w_i^{t-1})] = \mathbb{E}[\Delta R_i^t] \tag{9}$$

According to equation 9, it is easy to prove equation 8 is correct.

Lemma 2. Suppose we have N clients in our federated learning cluster, Algorithm 1 and Algorithm 2 can guarantee to recover local model on each round.

To reduce the amount of data being download, the server send compressed model $\overline{\Delta R^t}$ to client (see algorithm 1), whenever the client receive $\overline{\Delta R^t}$, we can not apply it directly, because we have to recover to local model G_i^t, the first step in algorithm 2 show how to

Table 1: Mnist data distribution of each federated client

client	total image number	label distribution									
		0	1	2	3	4	5	6	7	8	9
client 1	5832	763	586	423	533	582	607	690	568	539	541
client 2	5230	649	798	518	330	429	380	320	731	626	449
client 3	6190	363	724	722	421	611	612	531	669	708	829
client 4	6832	518	831	605	1082	637	607	996	586	701	269
client 5	5903	420	570	588	763	288	688	611	641	600	734
client 6	5598	518	763	563	597	591	531	328	608	421	678
client 7	6239	611	524	855	628	789	664	511	709	790	158
client 8	6166	417	449	511	666	737	508	1080	563	563	672
client 9	5298	523	960	739	707	432	498	285	420	488	246
client 10	6712	1141	537	434	404	746	326	566	770	425	1373

perform this process, the correctness can be guaranteed by the following equation:

$$\begin{aligned} G^0 + \sum_{j=1}^{n} \overline{R^j} &= G^0 + \sum_{j=1}^{k} \overline{R^j} - \sum_{j=1}^{k} \overline{R^j} + \sum_{j=1}^{n} \overline{R^j} \\ &= (G^0 + \sum_{j=1}^{k} \overline{R^j}) + (\sum_{j=1}^{n} \overline{R^j} - \sum_{j=1}^{k} \overline{R^j}) \\ &= G^k + \sum_{j=k+1}^{n} \overline{R^j} = G^n \end{aligned} \tag{10}$$

Since k and n are random, which satisfy $k \leq n$, using mathematical induction, we can easy prove the correctness.

3.6 Performance Analysis

The overall communication costs consist of two aspects: the upload from client to server and download from server to client.

- **Upload:** According to Algorithm 2, the total number of uploaded data is equal to:

$$S_{upload} = \sum_{j=1}^{M} |\Delta R_j^t| \tag{11}$$

  Since we execute spatial mean pooling on convolutional layer, suppose the kernel shape of model G_i^t is (n, s, s, m), where n is the number of filter input, m is the number of filter output size, s is kernel size. after compressed, the kernel shape of model ΔR_i^t is $(n, 1, 1, m)$. Obviously, compared with standard federated learning, the data upload amount reduce $(s * s)$ times for each convolutional kernel.
- **Download:** According to Algorithm 1, the total number of downloaded data is equal to:

$$S_{download} = |\overline{\Delta R^t}| \tag{12}$$

  $\overline{\Delta R^t}$ is the aggregation of all RPN model ΔR_i^t collected from selective clients, the shape of model $\overline{\Delta R^t}$ is equal to ΔR_i^t, as previously discussed, the data download amount also reduce $(s*s)$ times for each convolutional kernel.

4 Experiments

We evaluate our approach on three different tasks: image classification, object detection and semantic segmentation, we will compare the performance between standard federated learning and RPN. All our experiments are run on a federated learning cluster consists of one server side and 10 clients, each of which are equipped with 8 NVIDIA TeslaV100 GPUs.

4.1 Experiments on MNIST classification

We evaluate classification performance on mnist dataset using a 5-layers CNN model, which contains total 1,199,882 parameters, after using RPN technique, the transmitted data can drop to 1,183,242, since most of the parameters are came from full connection layer, the compression effect is not obvious enough. we split the data into 10 parts randomly, and send it to each client for training datasets, the data distribution of each client is shown in table 1.

The results are shown in Figure 4, as we can see, RPN can achieve almost the same effect as standard federated learning.

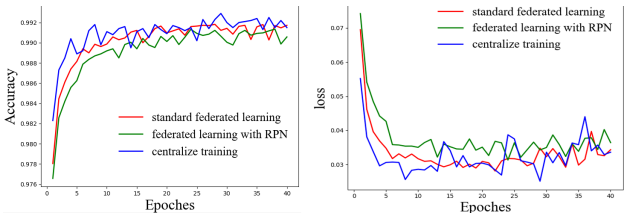

Figure 4: Experiments on MNIST dataset

Our experiment performance summarize in table 2:

Table 2: experiments performance on mnist classification

Model	MNIST	
	Parameters transmission	Accuracy
FL	1,199,882	0.991
RPN	1,183,242	0.9912
Performance	↓ **1.4%**	—

4.2 Experiments on Pascal-VOC object detection

In this section, we conduct object detection task on PASCAL VOC dataset. The PASCAL Visual Object Classification (PASCAL VOC)

Table 3: PASCAL VOC data distribution of each federated client

label	client id									
	client 1	client 2	client 3	client 4	client 5	client 6	client 7	client 8	client 9	client 10
Aeroplane	53	32	71	66	58	61	38	45	39	60
Bicycle	21	62	33	48	39	56	57	68	42	40
Bird	61	68	103	52	50	75	37	43	59	50
Boat	31	42	71	52	23	36	43	54	29	40
Bottle	71	82	38	54	65	98	37	77	61	81
Bus	31	12	53	64	55	46	27	18	59	60
Car	123	108	45	64	72	151	77	48	89	103
Cat	75	89	101	161	77	87	89	58	113	94
Chair	170	65	79	91	106	118	68	91	117	67
Cow	12	25	39	33	41	29	17	18	29	20
Diningtable	34	28	45	72	52	61	37	28	49	30
Dog	208	102	113	178	58	69	91	82	119	98
Horse	43	25	31	49	72	28	19	31	27	42
Motorbike	16	72	21	40	53	39	72	68	49	52
Person	340	689	213	480	111	239	128	196	295	303
PottedPlant	45	24	65	83	28	31	39	18	45	51
Sheep	28	12	61	22	14	20	30	18	39	27
Sofa	42	39	29	37	35	20	51	54	32	30
Train	34	26	66	14	25	32	38	40	19	41
Tvmonitor	68	43	23	45	39	29	17	29	31	40

dataset is a well-known dataset for object detection, classification, segmentation of objects and so on. There are around 10,000 images for training and validation containing bounding boxes with objects. Although, the PASCAL VOC dataset contains only 20 categories, it is still considered as a reference dataset in the object detection problem.

We conduct object detection task on pascal voc dataset with YOLOV3. YOLOV3 contrains 107 layer with 75 are convolutional layers. same as mnist experiment, we split the data into 10 parts randomly, and send it back to each client for training data. The data distribution of each client is shown in table 3.

Our experiment results are shown in Figure 5. We use mean average precision (mAP) as our performance metrics, mAP is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. As we can see, rpn will shuffe fluctuation in the early stage, but as the iterations continue, the performance becomes stable, and shows shallow performance gap between rpn and standard federated learning.

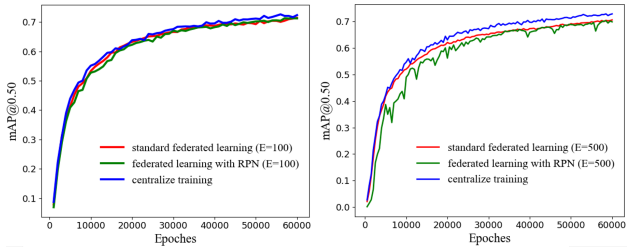

Figure 5: Experiments on PASCAL VOC object detection

Our experiment performance summarize in table 4, since in YOLOV3 network architecture, about 70% layers are convolutional layer, compared with mnist classification, our RPN compression strtegy is more obvious, we can reduce model parameters from 61,895,776 to 12,382,560, while only one percent of performance is declining.

Table 4: experiments performance on PASCAL VOC Object detection

Model	PASCAL VOC	
	Parameters transmission	Accuracy
FL	61,895,776	0.7077
RPN	12,382,560	0.7002
Performance	↓ **80%**	↓ **1%**

4.3 Experiments on Pascal-VOC Semantic segmentation

In this section, we conduct Semantic segmentation task on PASCAL VOC dataset, and use fully convolutional networks (FCN) as our model. FCN can use different model architecture for feature extraction, such as ResNet, vggnet, DenseNet, etc. In our experiment, we use Resnet50 as our based models. The data distribution of each client is shown in table 3.

Our experiment results are shown in Figure 6. We use meanIOU and pixel accuracy as our performance metrics.

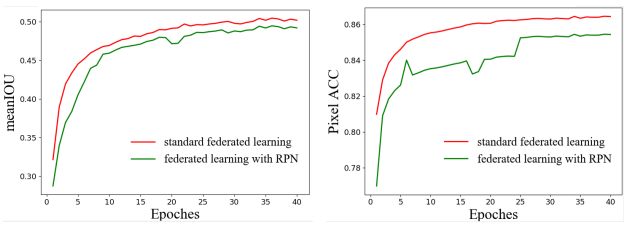

Figure 6: Experiments on PASCAL VOC semantic segmentation

Our experiment performance summarize in table 5, we can reduce model parameters from 23,577,621 to 13,325,281, while only 2.6 percent of performance is declining for meanIOU metric, and 1.16 percent of performance is declining for pixel accuracy.

Table 5: experiments performance on PASCAL VOC Object detection

Model	PASCAL VOC		
	Parameters transmission	meanIOU	Pixel ACC
FL	23,577,621	0.5077	0.864
RPN	13,325,281	0.4942	0.854
Performance	↓ **44%**	↓ **2.6%**	↓ **1.16%**

5 CONCLUSION and Future Work

In this paper, we propose a new compression strategy to improve communication efficiency of federated learning. we test our approach on three different tasks, including classification, object detection and semantic segmentation, the experiments show that RPN not only reduce data transmission effectively, but also achieve almost the same performance as compared to standard federated learning. Most importantly, RPN is n end-to-end procedure, which makes it easy to deploy in real-world application without human intervention. In the future, we will combine other compression strategies to improve communication efficiency.

REFERENCES

[1] Alham Fikri Aji and Kenneth Heafield, 'Sparse communication for distributed gradient descent', *arXiv preprint arXiv:1704.05021*, (2017).
[2] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar, 'signsgd: compressed optimisation for non-convex problems', *CoRR*, **abs/1802.04434**, (2018).
[3] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar, 'signsgd with majority vote is communication efficient and byzantine fault tolerant', *CoRR*, **abs/1810.05291**, (2018).
[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, 'Enriching word vectors with subword information', *CoRR*, **abs/1607.04606**, (2016).
[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander, 'Towards federated learning at scale: System design', *CoRR*, **abs/1902.01046**, (2019).
[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth, 'Practical secure aggregation for privacy-preserving machine learning', in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 1175–1191, New York, NY, USA, (2017). ACM.
[7] Sebastian Caldas, Jakub Konecný, H. Brendan McMahan, and Ameet Talwalkar, 'Expanding the reach of federated learning by reducing client resource requirements', *CoRR*, **abs/1812.07210**, (2018).
[8] Yudong Chen, Lili Su, and Jiaming Xu, 'Distributed statistical machine learning in adversarial settings: Byzantine gradient descent', *CoRR*, **abs/1705.05491**, (2017).
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, 'BERT: pre-training of deep bidirectional transformers for language understanding', *CoRR*, **abs/1810.04805**, (2018).
[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, The MIT Press, 2016.
[11] Song Han, Huizi Mao, and William J. Dally, 'Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding', in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, (2016).
[12] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng, 'Deep speech: Scaling up end-to-end speech recognition', *CoRR*, **abs/1412.5567**, (2014).
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Spatial pyramid pooling in deep convolutional networks for visual recognition', *CoRR*, **abs/1406.4729**, (2014).
[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', *CoRR*, **abs/1512.03385**, (2015).
[15] Sergey Ioffe and Christian Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *CoRR*, **abs/1502.03167**, (2015).
[16] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, 'Federated learning: Strategies for improving communication efficiency', *CoRR*, **abs/1610.05492**, (2016).
[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, (2012). Curran Associates Inc.
[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, 'Deep learning', *Nature*, **521**(7553), 436–444, (5 2015).
[19] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey, 2019.
[20] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas, 'Federated learning of deep networks using model averaging', *CoRR*, **abs/1602.05629**, (2016).
[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. cite arxiv:1301.3781.
[22] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, 'Robust and communication-efficient federated learning from non-iid data', *CoRR*, **abs/1903.02891**, (2019).
[23] Nikko Strom, 'Scalable distributed dnn training using commodity gpu cloud computing', in *Sixteenth Annual Conference of the International Speech Communication Association*, (2015).
[24] Ananda Theertha Suresh, Felix X. Yu, H. Brendan McMahan, and Sanjiv Kumar, 'Distributed mean estimation with limited communication', *CoRR*, **abs/1611.00429**, (2016).
[25] Zeyi Tao and Qun Li, 'esgd: Communication efficient distributed deep learning on the edge', in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, (July 2018). USENIX Association.
[26] Yusuke Tsuzuku, Hiroto Imachi, and Takuya Akiba, 'Variance-based gradient compression for efficient distributed deep learning', *arXiv preprint arXiv:1802.06058*, (2018).
[27] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu, 'Wavenet: A generative model for raw audio', *CoRR*, **abs/1609.03499**, (2016).
[28] Paul Voigt and Axel von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, Springer Publishing Company, Incorporated, 1st edn., 2017.
[29] Hongyi Wang, Scott Sievert, Zachary Charles, Shengchao Liu, Stephen Wright, and Dimitris Papailiopoulos. Atomo: Communication-efficient learning via atomic sparsification, 2018.
[30] Luping Wang, Wei Wang, and Bo Li, 'Cmfl: Mitigating communication overhead for federated learning', in *2019 IEEE 39th International Conference on Distributed Computing Systems*, Dallas, TX, (2019).
[31] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong, 'Federated machine learning: Concept and applications', *CoRR*, **abs/1902.04885**, (2019).
[32] Yu Zhang, William Chan, and Navdeep Jaitly, 'Very deep convolutional networks for end-to-end speech recognition', *CoRR*, **abs/1610.03022**, (2016).