EFFICIENT FEDERATED LEARNING VIA VARIATIONAL DROPOUT

Anonymous authors
Paper under double-blind review

ABSTRACT

As an emerging field, federated learning has recently attracted considerable attention. Compared to distributed learning in the datacenter setting, federated learning has more strict constraints on computate efficiency of the learned model and communication cost during the training process. In this work, we propose an efficient federated learning framework based on variational dropout. Our approach is able to jointly learn a sparse model while reducing the amount of gradients exchanged during the iterative training process. We demonstrate the superior performance of our approach on achieving significant model compression and communication reduction ratios with no accuracy loss.

1 INTRODUCTION

Federated Learning is an emerging machine learning approach that has recently attracted considerable attention due to its wide range of applications in mobile scenarios (McMahan et al., 2017; Konečnỳ et al., 2016; Smith et al., 2017). It enables geographically distributed devices such as mobile phones to collaboratively learn a shared model while keeping the training data on each phone. This is different from standard machine learning approach which requires all the training data to be centralized in a server or in a datacenter. As such, federated learning enables distributing the knowledge across phones without sharing users' private data.

Federated Learning uses some form of distributed stochastic gradient descent (SGD) and requires a parameter server to coordinate the training process. The server initializes the model and distributes it to all the participating devices. In each distributed SGD iteration, each device computes the gradients of the model parameters using its local data. The server aggregates the gradients from each device, averages them, and sends the averaged gradients back. Each device then updates the model parameters using the averaged gradients. In such manner, each device benefits from obtaining a better model than the one trained only on the locally stored private data.

While federated learning shares some common features with distributed learning in the datacenter setting (Dean et al., 2012; Li et al., 2014) since they both use distributed SGD as the core training technique, federated learning has two more strict constraints which datacenter setting does not have:

Model Constraint: Compared to datacenters, mobile devices have much less compute resources. This requires the *final* model learned in the federated learning setting to be computationally efficient so that it can efficiently run on mobile devices.

Communication Constraint: In datacenters, communication between the server and working nodes during SGD is conducted via Gbps Ethernet or InfiniBand network with even higher bandwidth (Wen et al., 2017). In contrast, communication in the federated learning setting relies on wireless networks such as 4G and Wi-Fi. Both uplink and downlink bandwidths of those wireless networks are at Mbps scale, which is much lower than the Gbps scale in the datacenter setting. The limited bandwidth in the federated learning setting illustrates the necessity of reducing the communication cost to accelerate the training process.

In this work, we propose an efficient federated learning framework that meets both model and communication constraints. Our approach is inspired by *variational dropout* (Kingma et al., 2015). Our *key idea* is to *jointly* and *iteratively sparsify* the parameters of the shared model to be learned as well as the gradients exchanged between the server and the participating devices during the distributed SGD training process. By sparsifying parameters, only important parameters are kept, and the final model learned thus becomes computationally efficient run on mobile devices. By sparsifying gradients, only important gradients are transmitted, and the communication cost is thus significantly reduced. We examine the performance of our framework on three deep neural networks and five datasets that fit the federated learning setting and are appropriate to be deployed on resource-limited mobile devices. Our experiment results show that our framework is able to achieve significant model compression and communication reduction ratios with no accuracy loss.

2 Related Work

Federated Learning. As an emerging field, federated learning has recently attracted a lot of attention. McMahan et al. (2017) developed a federated learning approach based on iterative model averaging to tackle the statistical challenge, especially for mobile setting. Smith et al. (2017) proposed a distributed learning framework based on multi-task learning and demonstrated their approach is able to address the statistical challenge and is robust to stragglers. Konečnỳ et al. (2016) developed structured and sketched update techniques to reduce uplink communication cost in federated learning. In comparison, our work – to the best of our knowledge – represents the first federated learning framework that achieves model sparsification and communication reduction in a unified manner.

Gradient Compression. Our work is related to gradient compression in distributed SGD. One line of research is to use low-bit gradient representation to reduce the gradient communication overhead. In Wen et al. (2017), the authors proposed TernGrad which only used three numerical levels to represent gradients. In Alistarh et al. (2017), the authors presented QSGD that allows dynamic trade-off between accuracy and communication cost by choosing bit width of gradients. As another line of research, some previous works aimed to reduce communication cost by transmitting only a small portion of gradients. For example, Konecnỳ & Richtárik (2016) proposed using the structured update to sparsify the gradients. Aji & Heafield (2017) and Lin et al. (2018) used magnitudes as indicators to measure the importance of gradients and proposed to only transmit gradients whose magnitudes are larger than a threshold. In our work, we use an additional dropout parameter as the importance indicator, and transmit gradients based on the values of the dropout parameters stochastically.

Model Sparsification. Our work is also related to model sparsification. Model sparsification aims to reduce the computational intensity of deep neural networks by pruning out redundant model parameters. In (Han et al., 2016), the authors developed a parameter pruning method that can sparsify deep neural networks by an order of magnitude without losing the prediction accuracy. The novelty of our proposed approach lies at integrating model sparsification and communication reduction into a single optimization framework based on training data distributed at different devices.

3 Preliminaries

Bayesian Neural Networks. Our work is inspired by Bayesian neural networks (Polson et al., 2017). Conventional neural networks learn scalar weights via maximum likelihood estimation. In comparison, a Bayesian neural network associates each weight with a distribution. Having a distribution instead of a single scalar value takes account of uncertainty for weight estimates and is thus more robust to overfitting (Hernández-Lobato & Adams, 2015).

Consider a Bayesian neural network parameterized by weight matrix $\mathbf{w}$. Let $p(\mathbf{w})$ denote the prior distribution of $\mathbf{w}$. Given a dataset $\mathcal{D}$ with N data samples (x_n, y_n) for $n \in [1, N]$, the posterior distribution of $\mathbf{w}$ is $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D})$. Unfortunately, $p(\mathbf{w}|\mathcal{D})$ is generally computationally intractable, and thus approximation approaches are needed. One popular approximation approach is *variational inference*, which uses a parametric distribution $q_\phi(\mathbf{w})$ to approximate the true posterior distribution $p(\mathbf{w}|\mathcal{D})$ (Kingma & Welling, 2013). The parameters ϕ of $q_\phi(\mathbf{w})$ are learned by maximizing the *variational lower bound* $\mathcal{L}(\phi)$ defined as follows (Kingma et al., 2015):

$$\mathcal{L}(\phi) = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{w})} \log p(y_n|x_n, \mathbf{w}) - D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})). \tag{1}$$

The first term on the right side measures the predictive performance of $q_\phi(\mathbf{w})$. The second term is the KL-divergence between $q_\phi(\mathbf{w})$ and $p(\mathbf{w})$, which regularizes $q_\phi(\mathbf{w})$ to be close to $p(\mathbf{w})$.

Variational Dropout. Dropout is one of the most effective regularization methods for training deep neural networks (Srivastava et al., 2014). In traditional dropout training, either Bernoulli or Gaussian

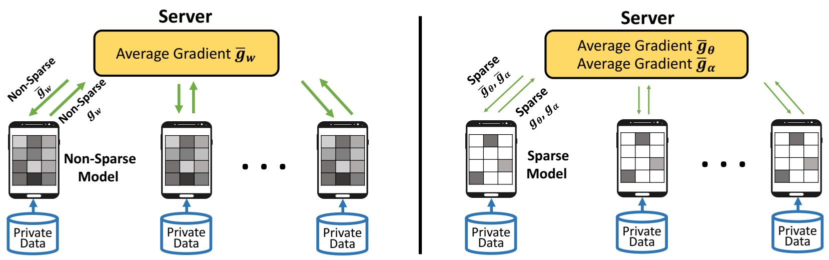

Server
Average Gradient $\overline{g}_w$
Non-Sparse $\overline{g}_w$
Non-Sparse g_w
Non-Sparse Model
Private Data
Private Data
Private Data
Server
Average Gradient $\overline{g}_\theta$
Average Gradient $\overline{g}_\alpha$
Sparse $\overline{g}_\theta, \overline{g}_\alpha$
Sparse g_θ, g_α
Sparse Model
Private Data
Private Data
Private Data

Figure 1: **Left:** The standard federated learning framework. The gradients transmitted from each device to the central server is *non-sparse*, and the shared model learned from the distributed private data is *non-sparse*. **Right:** The proposed efficient federated learning framework based on unified gradient and model sparsification. The gradients transmitted from each device to the central server is *sparse*, and the shared model learned from the distributed private data is *sparse*.

noises are added to the weights with a *fixed* dropout rate p (Srivastava et al., 2014). In (Kingma et al., 2015), the authors connected dropout with Bayesian neural networks, and showed that a *different* dropout rate p_{ij} can be learned for each individual weight w_{ij} in $\mathbf{w}$ if the Gaussian noise $\xi_{ij} \sim \mathcal{N}(1, \alpha_{ij} = p_{ij}/(1 - p_{ij}))$ is added. As a result, each weight w_{ij} in $\mathbf{w}$ has the following Gaussian distribution parameterized by $\phi_{ij} = (\theta_{ij}, \alpha_{ij})$:

$$w_{ij} \sim \mathcal{N}(\theta_{ij}, \alpha_{ij}\theta_{ij}^2) = q_{\phi_{ij}}(w_{ij}) \tag{2}$$

where $\alpha_{ij}\theta_{ij}^2$ is the variance of the Gaussian distribution, and α_{ij} is enforced to be greater than zero.

Given $q_{\phi_{ij}}(w_{ij})$ in (2), the first term in (1) can be approximated by Monte Carlo estimation (Kingma et al., 2015). To compute the second term in (1), the authors in (Kingma et al., 2015) adopted a prior distribution $p(\mathbf{w})$ which makes $D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w}))$ *only* depend on α_{ij}. As a result, the second term in (1) can be approximated by

$$D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})) \approx \sum_{i,j} \left(-0.64 * \sigma(1.87 + 1.49 * \log\alpha_{ij}) + 0.5 * \log(1 + \alpha_{ij}^{-1})\right) + C, \tag{3}$$

where C is a constant and σ is the sigmoid function $\dfrac{1}{1 + e^{-x}}$.

Sparsifying Bayesian Neural Networks via Variational Dropout. Given that variational dropout allows learning different dropout rates for different weights, the authors in (Molchanov et al., 2017) showed that variational dropout can be used to sparsify Bayesian neural networks by pruning the weights whose learned dropout rates are high while still achieving comparable predictive accuracies comparing to the unpruned ones. This is because a large dropout rate corresponds to large noise in the weight. The large noise causes the value of this weight to be random and unbounded, which will corrupt the model prediction. As a result, pruning this weight out would be beneficial to the prediction performance of the learned neural network.

Our work is inspired by this key finding. We adopt variational dropout to sparsify neural networks in the federated learning setting. We describe the details of our approach in the following section.

4 Our Algorithm

Standard Federated Learning Framework. Figure 1 (left) illustrates the standard federated learning framework based on synchronous SGD (we leave asynchronous SGD for future work). The framework consists of a parameter server and a number of geographically distributed devices. In each distributed SGD iteration, each device computes the gradients of the model parameters based on its local data, and sends the gradients (*non-sparse*) to the server. The server averages the gradients aggregated from every device and sends the averaged gradients (*non-sparse*) to all devices. Each device then uses the received averaged gradients to update its local model (*non-sparse*).

Algorithm 1 Efficient Federated Learning based on Variational Dropout.

```
Input:
    $N$ devices with local private datasets $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots, \mathcal{D}_N$.
    Base model parameter $\boldsymbol{\theta}^0$ , Dropout parameter $\boldsymbol{\alpha}^0$, Learning rate $\eta$, the threshold value T.
Output:
    A sparse model $\boldsymbol{w}$ shared by $N$ decentralized devices.

    Initialize: $\boldsymbol{\theta}_i = \boldsymbol{\theta}^0, \boldsymbol{\alpha}_i = \boldsymbol{\alpha}^0, i = 1, ..., N$.
 1: for each iteration do:
 2:     for $i = 1 : N$ do:
 3:         Device $i$ selects a mini-batch $b_i$ from local private dataset $\mathcal{D}_i$.
 4:         Device $i$ computes the sparse gradients $\boldsymbol{g}_{\boldsymbol{\theta}_i} = \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i \odot M(\boldsymbol{\alpha}_i)$,  $\boldsymbol{g}_{\boldsymbol{\alpha}_i} = \nabla_{\boldsymbol{\alpha}_i}\mathcal{L}_i \odot M(\boldsymbol{\alpha}_i)$,
            where $M(\boldsymbol{\alpha}_i)$ is a 0-1 binary matrix, and each entry $M(\alpha_{ij})$ is put to 0 if $\alpha_{ij}$ > T otherwise 1.
 5:         Device $i$ uploads $(\boldsymbol{g}_{\boldsymbol{\theta}_i}, \boldsymbol{g}_{\boldsymbol{\alpha}_i})$ to the central server.
 6:     The central sever receives the uploaded model parameter gradients and computes their average:
            $\bar{\boldsymbol{g}}_{\boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{g}_{\boldsymbol{\theta}_i}, \bar{\boldsymbol{g}}_{\boldsymbol{\alpha}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{g}_{\boldsymbol{\alpha}_i}$.
 7:     for $i = 1 : N$ do:
 8:         Device $i$ pulls averaged gradients from the central server and updates parameters:
                $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta \cdot \bar{\boldsymbol{g}}_{\boldsymbol{\theta}}, \boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i - \eta \cdot \bar{\boldsymbol{g}}_{\boldsymbol{\alpha}}$.
```

Given that the standard federated learning framework suffers from both model and communication constraints described in the introduction section, we propose an efficient federated learning framework (Figure 1 (right)) that meets both model and communication constraints.

Our Algorithm. Algorithm 1 formulates our proposed federated learning framework based on variational dropout. Let N denote the number of devices, and $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, ..., \mathcal{D}_N$ be the dataset stored inside the N devices, respectively. The goal of our proposed federated learning framework is to have all these N devices to collectively learn a sparse model at the final stage while reducing the communication cost during the training process.

Before the training process starts, the server initializes a base model parameterized by ($\boldsymbol{\theta}^0$, $\boldsymbol{\alpha}^0$), sets the learning rate to be η, and distributes the base model and the learning rate to all the N devices. As such, all the devices have the same initial parameters to start with. In practice, $\boldsymbol{\alpha}^0$ are initialized with random values; $\boldsymbol{\theta}^0$ can be initialized with either random values or the pretrained weights from other dataset such as ImageNet (Deng et al., 2009).

During each distributed SGD iteration, the i^{th} device first selects a mini-batch b_i from its local private dataset $\mathcal{D}_i$, and computes the gradients of the loss function in (1), including model parameter gradient $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i$ and dropout parameter gradient $\nabla_{\boldsymbol{\alpha}_i}\mathcal{L}_i$, where $\boldsymbol{\theta}_i, \boldsymbol{\alpha}_i$ are the parameters of the i^{th} device and $\mathcal{L}_i$ is the *variational lower bound* of the i^{th} device.

The iterative exchange of gradients $(\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i, \nabla_{\boldsymbol{\alpha}_i}\mathcal{L}_i)$ would incur massive communication overhead and becomes the communication bottleneck in standard federated learning framework. To reduce the communication overhead, our *key idea* is to utilize the dropout parameters $\boldsymbol{\alpha}$ to identify important gradients to transmit to the server (Line 4). As explained in the preliminaries section, a large dropout rate corresponds to large noise in the weight. The large noise causes the value of this weight to be random and unbounded, which will corrupt the model prediction. As a result, pruning this weight out would be beneficial to the prediction performance of the learned neural network. Recall $\alpha_{ij} = p_{ij}/(1 - p_{ij})$. A large dropout rate p_{ij} leads to a large dropout parameter α_{ij}. As such, the dropout parameter α_{ij} can be viewed as an *importance indicator* of its corresponding model parameter θ_{ij}, where higher value α_{ij} corresponds to less important θ_{ij}.

Inspired by this property, the i^{th} device is able to identify important gradients $(\boldsymbol{g}_{\boldsymbol{\theta}_i}, \boldsymbol{g}_{\boldsymbol{\alpha}_i})$ by:

$$\begin{aligned} \boldsymbol{g}_{\boldsymbol{\theta}_i} &= \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i \odot M(\boldsymbol{\alpha}_i) \\ \boldsymbol{g}_{\boldsymbol{\alpha}_i} &= \nabla_{\boldsymbol{\alpha}_i}\mathcal{L}_i \odot M(\boldsymbol{\alpha}_i) \end{aligned} \tag{4}$$

where $M(\boldsymbol{\alpha}_i)$ is a (0,1)-matrix and $\odot$ represents *Hadamard product*. Each entry $M(\alpha_{ij})$ in $M(\boldsymbol{\alpha}_i)$ is put to 0 if α_{ij} > T otherwise 1.

As shown in the experiment section, $M(\boldsymbol{\alpha}_i)$ will become highly sparse during the training process and thus $(\boldsymbol{g}_{\boldsymbol{\theta}_i}, \boldsymbol{g}_{\boldsymbol{\theta}_i})$ will also be highly sparse. *It should be emphasized that although we upload a pair of sparse gradients ($\boldsymbol{g}_{\boldsymbol{\theta}_i}$, $\boldsymbol{g}_{\boldsymbol{\alpha}_i}$) (Figure 1 (right)) compared to non-sparse gradients $\boldsymbol{g}_{\boldsymbol{w}_i}$ in the standard federated learning framework (Figure 1 (left)), due to highly sparse ($\boldsymbol{g}_{\boldsymbol{\theta}_i}$, $\boldsymbol{g}_{\boldsymbol{\alpha}_i}$), the total number of exchanged gradients is significantly reduced.*

The remaining steps are the same with the standard federated learning framework except that we upload $\boldsymbol{g}_{\boldsymbol{\alpha}_i}$ and download $\bar{\boldsymbol{g}}_{\boldsymbol{\alpha}}$. The server averages the uploaded gradients denoted as $(\bar{\boldsymbol{g}}_{\boldsymbol{\theta}}, \bar{\boldsymbol{g}}_{\boldsymbol{\alpha}})$ and sends back to the devices. Finally, each device downloads the averaged parameter gradients from the server and updates the model parameters with the preset learning rate η.

Algorithm Variants. During our experiments, we have observed two key insights. First, even though $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are independent variables, $\boldsymbol{\alpha}$ is empirically negatively correlated with $\boldsymbol{\theta}$, which is consistent with the observation in Molchanov et al. (2017). Second, θ_{ij} is suppressed by large α_{ij} and it can hardly grow back unless D_{KL} in (1) is removed. Based on these observations, we make some variants from Algorithm 1 to further reduce the communication cost.

Specifically, each device is forced to optimize $\boldsymbol{\alpha}$ locally. It neither uploads the gradients of $\boldsymbol{\alpha}$ to the server nor downloads the gradients of $\boldsymbol{\alpha}$ from the server. The rationale behind this strategy is that since $\boldsymbol{\alpha}$ is associated with $\boldsymbol{\theta}$ which is synchronized every iteration during SGD, $\boldsymbol{\alpha}$ will thus be forced to be almost the same across all the devices. Thus we can only need to upload gradients of important model parameters $\boldsymbol{g}_{\boldsymbol{\theta}}$ and omit the gradient of the dropout parameters.

In our experiments, we have implemented both Algorithm 1 and the variants described above. Our experimental results on variants show better performance on communication cost reduction while achieving convergence and the same test accuracies as Algorithm 1 and in our experimental section we only show the results on the algorithm variants. The experimental results on Algorithm 1 is shown in the supplementary material.

5 Experiments

5.1 Experimental Setup

Datasets and Deep Neural Networks. We evaluate the performance of our framework on three deep neural networks and five datasets that fit the federated learning setting. Specifically, we examined CifarNet (Krizhevsky et al., 2012) on MNIST (LeCun et al., 1998) and MNIST-Fashion (Xiao et al., 2017) datasets; ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky & Hinton, 2009) and Traffic Sign (tra) datasets, and VGG-like (Simonyan & Zisserman, 2015) on SVHN (Netzer et al., 2011) dataset, where VGG-like is obtained by removing the fully connected layers of the standard VGG-16 (Simonyan & Zisserman, 2015).

Protocol. We use the standard distributed learning framework illustrated in Figure 1 (left) as the baseline. We implement the baseline and our proposed framework using PyTorch (Paszke et al., 2017) and conduct experiments on NVIDIA 1080Ti GPUs. All the experimental settings are the same for the baseline and our proposed framework. In all experiments, we set the mini-batch size to 128 and used Adam (Kinga & Adam, 2015) as the SGD optimizer. Each dataset is randomly and evenly divided into N non-overlapping parts for N devices. To achieve state-of-the-art accuracy, all the deep neural networks were pre-trained on ImageNet (Deng et al., 2009) before they are trained on each of the five datasets. This setup also emulates the practical use scenario of federated learning in real-world settings where devices are loaded with pre-trained models before they participate in federated learning when new data comes in. For fair comparison, we run both the baseline and our proposed framework to full convergence on each dataset.

5.2 Results

5.2.1 Accuracy and Model Sparsity

We first examine the Top-1 test accuracy and the model sparsity achieved by our framework when 4 devices are included. Table 1 lists the top-1 accuracy achieved by the baseline (i.e., non-sparse) and our framework (i.e., sparse) as well as the the model sparsity (defined as the percentage of non-zero weights) for each dataset. Overall, our framework is able to sparsify the model with 1.4% to 7.2% non-zero weights across three different models trained on five different datasets. The sparse models have achieved comparable accuracy compared to the non-sparse ones.

5.2.2 Communication Cost Reduction

In the second set of experiments, we demonstrate the benefits of our framework in reducing communication costs during training. Figure 2 plots the communication cost curves of both the baseline (red dashed line) and our framework (blue solid line) during training on five datasets. The horizontal axis

Table 1: Top-1 accuracy and model sparsity of 4 devices.

Network	**Dataset**	**Params**	**Top-1 Accuracy (Non-Sparse)**	**Top-1 Accuracy (Sparse)**	**Percentage of Non-Zero Weights**
CifarNet	MNIST	1.3M	99.48%	99.31%	**2.9%**
CifarNet	MNIST-Fashion	1.3M	91.60%	91.51%	**7.2%**
ResNet-18	CIFAR-10	11.7M	93.57%	93.38%	**3.8%**
ResNet-18	Traffic Sign	11.7M	98.42%	98.18%	**1.4%**
VGG-like	SVHN	18.8M	95.30%	95.33%	**3.0%**

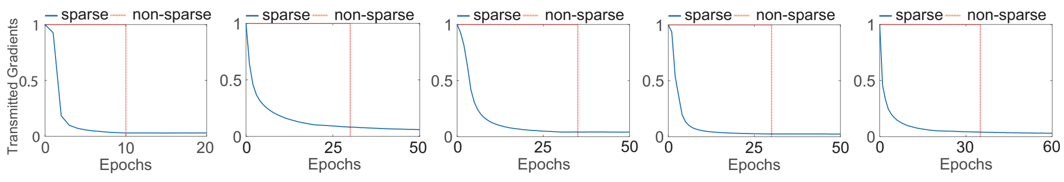
represents the epoch number during training. The vertical axis represents the percentage of gradients transmitted during both uploading and downloading stages.

We have two observations from the results. First, our framework takes more epochs than the baseline to converge. For example, as shown in Figure 2 (a), it takes 20 epochs for our framework to converge while it takes 10 epochs for the baseline. This is because compared to sending all the gradients during each SGD iteration, sending a sparse version of the gradients leads to a larger variance. Second, the percentage of gradients transmitted drops very quickly at the beginning of the training process. This indicates that it only takes a small number of SGD iterations for our framework to learn a model with a significantly high sparsity. In other words, it demonstrates the efficiency of our framework on learning sparse models in the federated setting.

Given the fact that our approach takes more epochs to converge, the key question we need to answer is whether our approach can reduce the communication costs by enough to offset the overhead caused by the extra epochs. To answer this question, Table 2 lists the data communication volumes of the baseline (i.e., non-sparse) and our approach (i.e., sparse) for each dataset. The data communication volume is the *total* transmission data volume (including both uploading and downloading stages) accumulated from all the SGD iterations until convergence during training. The value of the data communication volume can be seen as the area under the communication cost curves depicted in Figure 2. As shown in Table 2, our approach not only reduces the communication costs by enough to offset the overhead caused by the extra epochs, but also cuts the communication costs to 12.2% to 22.2% of the non-sparse framework across datasets. The success in reducing the communication costs by this large is attributed to our approach's fast sparsification speed illustrated in Figure 2. In fact, the transmission of sparse gradients include both gradient value itself and its corresponding index. However, the corresponding index overhead is negligible due to the high sparsity, and we omit it in calculating the communication cost (Abadi et al., 2016).

Table 2: Data communication volume and percentage of communication cost with 4 devices.

Network	**Dataset**	**Data Communication Volume (Non-Sparse)**	**Data Communication Volume (Sparse)**	**Percentage of Communication Cost**
CifarNet	MNIST	11.5 GB	2.1 GB	**18.2%**
CifarNet	MNIST-Fashion	34.5 GB	7.6 GB	**22.2%**
ResNet-18	CIFAR-10	297.5 GB	55.2 GB	**18.5%**
ResNet-18	Traffic Sign	200.3 GB	24.6 GB	**12.3%**
VGG-like	SVHN	702.2 GB	84.9 GB	**12.0%**



(a) MNIST (b) MNIST-Fashion (c) CIFAR-10 (d) Traffic Sign (e) SVHN

Figure 2: Communication cost curves during training on five datasets. The horizontal axis represents the epoch number during training. The vertical axis represents the percentage of gradients transmitted.

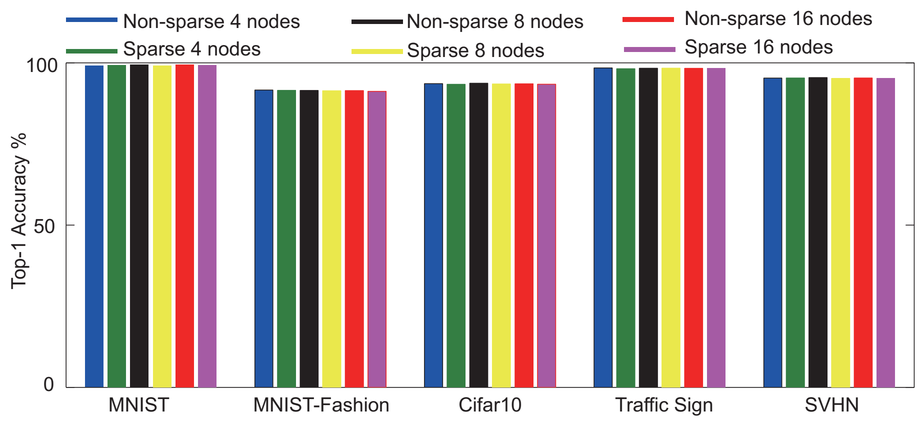
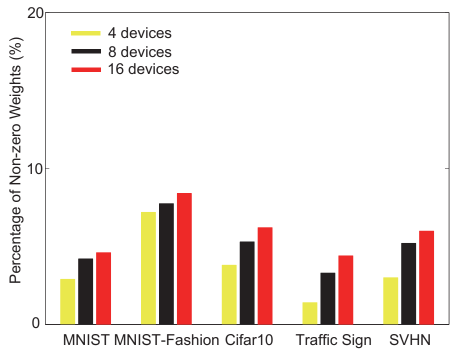
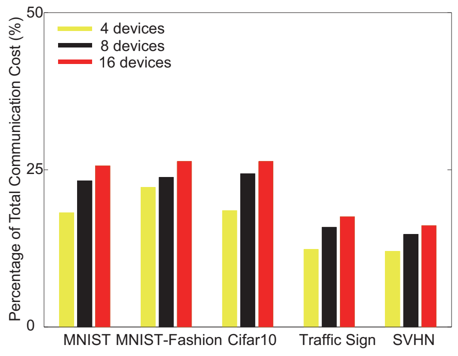


Figure 3: The top-1 accuracy of 4, 8 and 16 devices for non-sparse distributed learning and sparse federated learning.



(a)



(b)

Figure 4: (a) The percentage of non-zero weights of the model after training for 4, 8, and 16 devices; (b) the percentage of communication cost for 4, 8 and 16 devices.

5.2.3 Scalability Analysis

In this experiment, we examine the scalability of our framework when extending to larger number of devices. Figure 3 shows the performance of accuracy when extending to 8 and 16 devices, and Figure 4 shows the percentage of non-zero weights, and the percentage of communication cost when extending to 8 and 16 devices. We have the following observations from the results. First, our framework doesn't incur any accuracy loss for 8 and 16 devices, demonstrating its robustness across different models and datasets. Second, we observe that our framework scales well when the number of devices increases to 8 and 16. Specifically, with 8 devices, our framework achieves 3.3% to 7.7% non-zero weights and and cuts the communication costs to 14.7% to 24.3% compared to the non-sparse model. With 16 devices, our framework achieves 4.4% to 8.4% non-zero weights and cuts the communication costs to 16.1% to 26.3% compared to the non-sparse model.

We also observe that both percentage of non-zero weight and communication reduction increase slightly when the number of devices increases. This is due to the way we set up our experiments in which we randomly and evenly divided each dataset into N non-overlapping parts for N devices. As the number of devices increases, the data size allocated to each device decreases. As a consequence, the data distribution at each device becomes less similar, which makes it challenging to learn a global shared model with the same high sparsity without accuracy loss. In our experiments, we focused on maintaining the accuracy and thus compromised with a slightly increased percentage of non-zero weights and communication reduction.

5.2.4 Performance Analysis from Random Initialization

In previous experiments, the model parameters $\boldsymbol{\theta}$ are pretrained with ImageNet, and in this section we will display the performance of our proposed framework with random initialization weights. Table 3, Table 4 and Table 5 are the accuracy, percentage of non-zero weights and percentage of communication cost for 4, 8, and 16 devices with random initialization weights, respectively. Firstly, we see that the accuracy has decreased 2-3% compared to models using the pretrained weights. Secondly, the percentage of the non-zero weights and total communication cost has decreased accordingly compared to the model with pretrained weights. Starting from the random initialization has some drawbacks for the Bayesian dropout training since lots of weights will be pruned away in the early stage (Molchanov et al., 2017), before the model could possibly learn some useful representations of the data. As a result, the model sparsity will be higher while the accuracy will be lower. At the same time, the total communication cost will be lower compared to pretrained model due to the higher model sparsity.

Table 3: Top-1 accuracy of 4, 8, and 16 devices for random initialization weights.

Network	**Dataset**	**4 devices**	**8 devices**	**16 devices**
CifarNet	MNIST	98.35%	98.19%	98.23%
CifarNet	MNIST-Fashion	89.46%	89.51%	89.27%
ResNet-18	CIFAR-10	90.32%	90.19%	90.29%
ResNet-18	Traffic Sign	96.31%	96.36%	96.33%
VGG-like	SVHN	92.19%	92.18%	92.21%

Table 4: Percentage of non-zero weights of 4, 8, and 16 devices for random initialization weights.

Network	**Dataset**	**4 devices**	**8 devices**	**16 devices**
CifarNet	MNIST	1.9%	2.8%	3.2%
CifarNet	MNIST-Fashion	4.3%	5.4%	5.8%
ResNet-18	CIFAR-10	2.4%	3.5%	4.1%
ResNet-18	Traffic Sign	1.1%	1.8%	2.3%
VGG-like	SVHN	1.3%	2.0%	2.7%

Table 5: Percentage of communication cost of 4, 8, and 16 devices for random initialization weights.

Network	**Dataset**	**4 devices**	**8 devices**	**16 devices**
CifarNet	MNIST	11.6%	13.7%	15.2%
CifarNet	MNIST-Fashion	12.1%	12.6%	13.7%
ResNet-18	CIFAR-10	11.4%	13.3%	14.2%
ResNet-18	Traffic Sign	7.3%	9.7%	10.5%
VGG-like	SVHN	7.0%	8.7%	9.9%

6 Conclusion and Future Work

In this paper, we have presented a novel federated learning framework based on variational dropout for efficiently learning deep neural networks from distributed data. Our experiments on diverse deep network architectures and datasets show that our framework achieves high model sparsity with 2.9% to 7.2% non-zero weights and cuts the communication costs from 12.0% to 18.2% with no accuracy loss. Our experiments also show that our framework scales well when the number of devices increases. While this work focuses on the synchronous distributed SGD setting, we plan to examine the performance of our framework in the asynchronous distributed SGD setting as our future work. We also plan to explore the potential of our framework in further compressing the model and reducing the communication cost by combining with the orthogonal gradient quantization approaches.

7 Acknowledgement

We are grateful to NVIDIA Corporation for the donation of GPUs to support our research.

References

`http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset.`

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1707–1718, 2017.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

D Kinga and J Ba Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2013.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Jakub Konecnỳ and Peter Richtárik. Randomized distributed mean estimation: Accuracy vs communication. *arXiv preprint arXiv:1611.07555*, 2016.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pp. 583–598, 2014.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on LearningRepresentations (ICLR)*, 2018.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5, 2011.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Nicholas G Polson, Vadim Sokolov, et al. Deep learning: A bayesian perspective. *Bayesian Analysis*, 12(4):1275–1304, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4427–4437, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pp. 1508–1518, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A TOP-1 ACCURACY COMPARISON OF DIFFERENT TRAINING SETTINGS

In this section, we compare the performance of Algorithm 1 with different settings. The first setting is shown as our experiment section that model using the pretrained weights from Imagenet. The second setting is that we train our neural network from random initialization weights which is also shown in experiment section. The third setting is that we share the gradients of α using the pretrained weights. The accuracy performance comparison of these settings for 4, 8 and 16 devices are summarized in Table 6, Table 7 and Table 8. To be noted here, in our experimental setting, we don't not share α in order to further reduce communication cost for both the pretrained model and random initialized model. We have two observations. Firstly, the accuracy of the setting that shares α using the pretrained weights has the same accuracy of the setting using the pretrained weights without sharing α. Secondly, if the model is initialized with random weights, the accuracy will drop 2-3% compared to other two settings using pretrained weights.

Table 6: The accuracy comparison of 4 devices

Network	**Dataset**	**Pretrained weights without sharing α**	**Pretrained weights share α**	**Random initial weights**
CifarNet	MNIST	99.31%	99.35%	98.35%
CifarNet	MNIST-Fashion	91.51%	91.49%	89.46%
ResNet-18	CIFAR-10	93.38%	93.41%	90.32%
ResNet-18	Traffic Sign	98.18%	98.16%	96.31%
VGG-like	SVHN	95.33%	95.28%	92.19%

Table 7: The accuracy comparison of 8 devices

Network	**Dataset**	**Pretrained weights without sharing α**	**Pretrained weights share α**	**Random initial weights**
CifarNet	MNIST	99.37%	99.23%	98.19%
CifarNet	MNIST-Fashion	91.39%	91.36%	89.51%
ResNet-18	CIFAR-10	93.48%	93.45%	89.49%
ResNet-18	Traffic Sign	98.35%	98.29%	96.36%
VGG-like	SVHN	95.20%	95.38%	92.18%

Table 8: The accuracy comparison of 16 devices

Network	**Dataset**	**Pretrained weights without sharing α**	**Pretrained weights share α**	**Random initial weights**
CifarNet	MNIST	99.41%	99.33%	98.23%
CifarNet	MNIST-Fashion	91.42%	91.47%	89.27%
ResNet-18	CIFAR-10	93.52%	93.34%	90.29%
ResNet-18	Traffic Sign	98.29%	98.38%	96.33%
VGG-like	SVHN	95.31%	95.29%	92.21%

B PERCENTAGE OF NON-ZERO WEIGHTS COMPARISON OF DIFFERENT TRAINING SETTINGS

In this section, we report the percentage of non-zero weights for these three settings described above, and the results are shown as Figure 5.

We discover that the percentage of non-zero weights of the setting that shares α using the pretrained weights has the same level with the one uisng pretrained weights without sharing α. If we train the model from random initialized weights, the percentage of non-zero weights is lower than the two other settings, it means that we will obtain a more sparse model after training. However, the model accuracy will also be lower compared to other two settings.

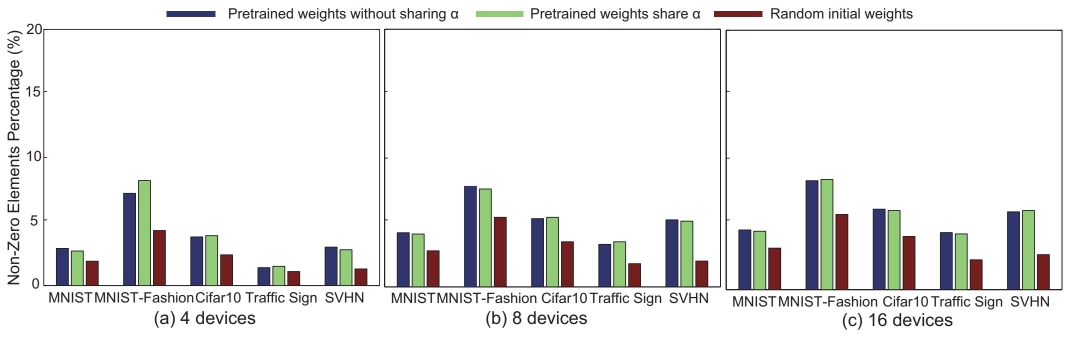


Figure 5: The percentage of non-zero weights of 4, 8, and 16 devices for sparse federated learning with pretrained weights, random weights and pretrained weights with shared α.

C Percentage of Total Communication Comparison of Different Training Settings

In this section, we report the total communication percentage of these three settings described above, and the results are shown as Figure 6.

First, we find that if the pretraied model shares α during the training, then the total communication percentage is nearly twice as much as the pretrained model without sharing. The communication cost is doubled because we share a pair of gradients $(\boldsymbol{g}_{\boldsymbol{\theta}_i}, \boldsymbol{g}_{\boldsymbol{\alpha}_i})$ compared to sharing $\boldsymbol{g}_{\boldsymbol{\theta}_i}$ only. Second, if we train our model from random initialized weights, the total communication cost in the minimal among the three settings. It is reasonable because the percentage of non-zero weights from random initialized weights is the lowest and thus the total communication exchange during training is also the lowest.

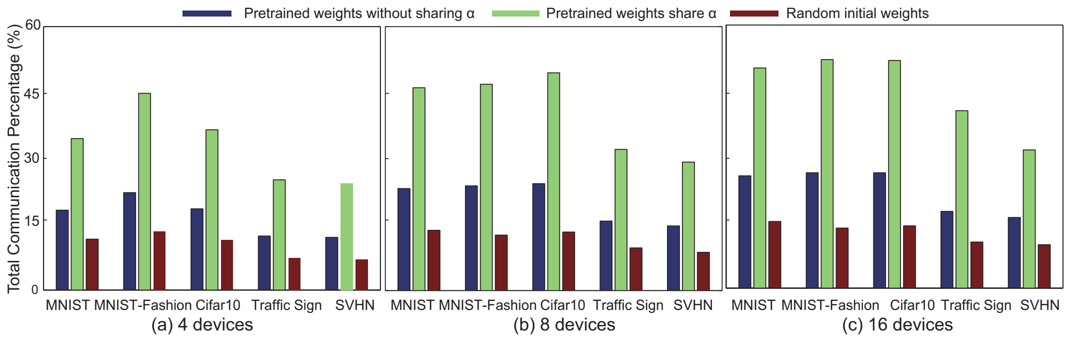


Figure 6: The total communication percentage of 4, 8, and 16 devices for sparse federated learning with pretrained weights, random weights and pretrained weights with shared α.