CE-SGD: Communication-Efficient Distributed Machine Learning

Zeyi Tao*, Qi Xia*, Qun Li* and Songqing Cheng†
* Department of Computer Science
William & Mary. Williamsburg, VA 23185
† Department of Computer Science
Geroge Mason University Fairfax, VA 22030

***Abstract*—Training large-scale machine learning models usually demands a distributed approach to process the huge amount of training data efficiently. However, the high network communication cost introduced by parallel stochastic gradient descent (SGD) algorithms is a well-known bottleneck. To this end, we propose CE-SGD, a communication-efficient distributed machine learning algorithm that aggressively reduces the amount of gradient data exchanged among the training workers. CE-SGD belongs to the family of gradient sparsification schemes. CE-SGD adaptively adjusts the gradient sparsity according to the model's feedback. It also selectively transmits the gradients based on their degree of participation in the backpropagation. We mathematically prove the convergence of CE-SGD for both convex and non-convex cases and conduct a series of experiments on our CE-SGD implementation. Our experiments reveal that CE-SGD can achieve fast convergence, desirable gradient compression ratio, and high accuracy with low network bandwidth cost compared to state-of-the-art algorithms.**

I. Introduction

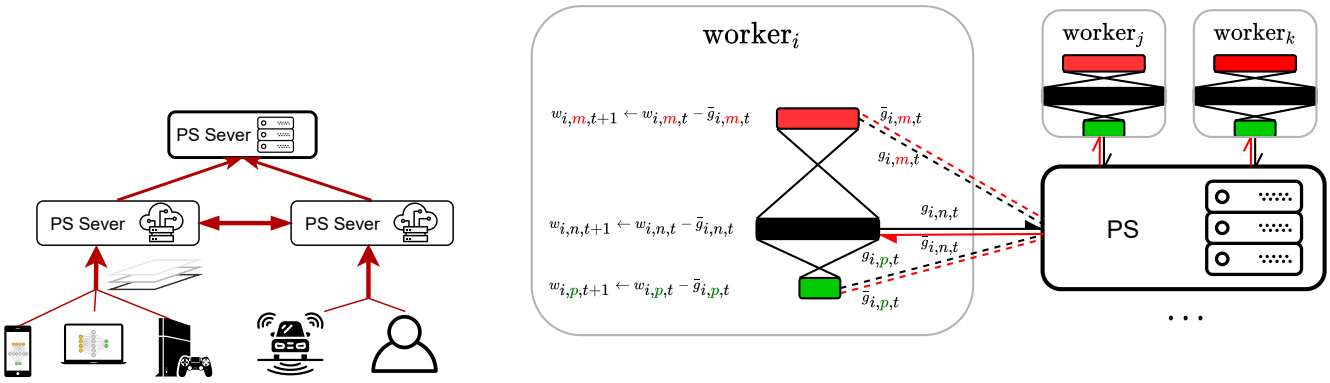
The data explosion mainly stimulated the remarkable proliferation of artificial intelligence (AI) in recent years. The emerging of fifth-generation (5G) wireless communication systems, at the same time, accelerates global data traffic growth. Given the ubiquitous smart mobile gadgets and Internet of Things (IoT) devices, they can easily access a wealth of data suitable for intelligence applications, which in turn can significantly improve users' daily lives. It is expected that a majority of intelligent applications will be deployed at the edge of wireless networks. Recently, the distributed learning frameworks [1], [2] aim at enabling the wireless devices to build a shared learning model collaboratively. In a central parameter server (PS) based machine learning paradigm, each worker retains an identical model copy and trains the model in parallel by feeding it with different subsets of data. The PS performs gradient synchronization, which collects the gradients from the synchronous stochastic gradient descent (SGD) algorithm from all the workers. The averaged gradient is then broadcast from the PS back to each worker to update their model parameters. The PS based approaches scale up distributed machine learning by increasing training workers and computational power.

However, a subsequent challenge emerges that the persistent gradient synchronization requires an enormous amount of network bandwidth [3]. Given the number of total workers N, the total communication cost for one round of training can be formulated as

$$2 \cdot N \cdot \mathcal{S} \tag{1}$$

where $\mathcal{S}$ is the size of the training model, including all weights and training metadata. For simplicity, we denote $\mathcal{S} = P \cdot$bit where P is the number of total trainable parameters and bit is precision such as 4, 8, 16. For example, as the winner of ILSVRC-2013 competition, AlexNet [4] comes along with nearly 61 million 32-bit real value parameters with an actual model size of $61 \times 10^6 \times 32 \approx 244$MB. The communication cost for one round of training is approximately 500Mb. The current averaged 5G speed is around 50Mbps. It will take approximately 10 sec to complete single iteration communication, significantly increasing the training time usage because training AlexNet or other typical deep neural networks (DNNs) usually requires 100k to 500k iterations.

To reduce the communication cost between training workers and the PS, two types of distributed SGD schemes have been widely studied: *gradient quantization* and *gradient sparsification*. Gradient quantization is quantizing the gradients to low precision or approximation value. For example, Quantized SGD (QSGD) [5] randomly quantizes gradient values by using uniformly distributed quantization levels to achieve a communication cost of $32 + d\log_2(2l+1)$ bits per communication round where d is the dimension of gradient and $l = 4, 8, 16$ is the predefined quantization level. Other approaches [6], [7], [8] are similar but different in error compensation mechanism. One drawback of gradient quantization is that it increases the stochastic variance and causes slow convergence. Gradient sparsification drops the less beneficial coordinates of gradient and then synchronizes with the PS to ensure unbiasedness. Gradient sparsification works due to two reasons. First, most deep neural models are over-parameterized [9], in which a considerable amount of model parameters have values close to zero. Therefore, we can use gradient sparsity to reduce the communication cost by only sending non-zero gradient coordinates to the PS. Second, even though the gradient is non-zero but small enough, we can accumulate them and send them to the PS for later synchronization. The above procedure can also be described as a variant of asynchronous SGD [10], [11] where the convergence is guaranteed. Even though gradient sparsification works in theory, it is impractical in real scenarios



(a) (b)

Fig. 1: (a) illustrates a typical PS based learning scheme. Model train and inference happen on local works. To enable model training, it only requires gradient communication between the PS and workers. (b) illustrates a detailed PS based learning model training process via synchronous SGD. Layer-wise model parameter $w_{i,m,t}$ is updated via averaged gradient $\bar{g}_{i,m,t}$.

because it is very hard to choose threshold values for dropping gradients in different layers of DNNs. For example, in our experiments, we observed gradient sparsification approach fails to converge if the thresholds are not chosen appropriately.

To address the issues mentioned above in gradient quantization and sparsification, we propose CS-SGD, a communication efficient distributed learning algorithm that allows sparse gradients to synchronize on the PS and aggressively reduces the overall communication costs. The main idea of CE-SGD is to select gradients for transmission based on the feedback from the parameter server. If a gradient has contributed to loss reduction (observed by the parameter server) multiple times, it will be selected for transmission with a higher probability. Otherwise, a gradient will not be transmitted. Aside from being communication efficient, CE-SGD has several other advantages:

- The implementation of CE-SGD is simple and is compatible with all existing training frameworks.
- CE-SGD can adaptively tune layer-wise gradient drop ratio for each layer without any threshold tuning.
- CE-SGD allows the model to choose the best gradients to improve the training speed and accuracy.
- Moreover, CE-SGD reduces the staleness effect and gains the critical model performance improvements by speculative gradient update.

II. Preliminaries

Notations Given a vector $\mathbf{x} \in R^k$ we denote its j-th entry by $\mathbf{x}^{(j)}$. We use $|\mathbf{x}^{(j)}|$ to denote the absolute value of $\mathbf{x}^{(j)}$. The $||\mathbf{x}||_2$ and $||\mathbf{x}||_\infty$ are l_2-norm and the maximum absolute entry of vector $\mathbf{x}$ respectively. We use $g_{i,m,t}$ to denote the m-th layer gradient that computed by i-th worker at iteration t. We use $g^{(e)}$ to denote the e-th entry of gradient g. Given two vectors $\mathbf{x}, \mathbf{y} \in R^k$, we use $\langle \mathbf{x}, \mathbf{y} \rangle$ to denote their inner product, $\mathbf{x} \odot \mathbf{y}$ to denote the element-wise product. This is also known as Hadamard product.

A. PS based learning and layer-wise synchronization

Figure 1(a) and Figure 1(b) illustrate a typical PS based deep learning model training process via synchronous SGD. At iteration t, a mini-batch of local training samples are fed to each worker i ($i \in [N]$). Worker i computes the gradient $g_{i,m,t}$ for deep learning model layer by layer *w.r.t* its input samples D_i. All gradients from the same model layer but on different local workers at iteration t are synchronized and averaged at the PS and then sent back to local workers. Frequently exchanging layer-wise gradients therefore causes huge amount of communication traffic.

B. Problem Formulation

We consider distributed learning with a central PS and N (N is larger) workers. Given a universal dataset D, the training tasks aim to solve the following minimization problem in a synchronous manner, also known as empirical risk minimization (ERM):

$$\arg\min_{\mathbf{x} \in \mathcal{R}^k} f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} E_{d \sim D_i}[\mathcal{F}_i(\mathbf{x}; d)] + \lambda r(\mathbf{x}) \quad (2)$$

where $\mathbf{x} \in R^k$ is the parameter we want to learn from the model. And $\mathcal{F}(\cdot;\cdot)$ is a prespecified loss function which measures the accuracy of the predictions and $r(\mathbf{x})$ is a regularization function. For simplicity, we let $f_i(\mathbf{x}) = E_{d \sim D_i}[\mathcal{F}_i(\mathbf{x}; d)]$ in the rest of the sections. Generally, this minimization problem can be solved by using stochastic gradient descent (SGD) algorithm, that is, with some proper initialization parameter x, we update the parameters as following rule:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t g_t \quad (3)$$

where $g_t = \nabla f(\mathbf{x}_t; d)$ is the gradient of the loss function at the current parameter $\mathbf{x}_t$ over a random data sample d from D_i and γ_t is current step size (learning rate).

Algorithm 1 Generalized Communication Efficient Algorithm

Initialization Initialize model parameter x_i for all workers. $g_{i,t}$ is local gradient on worker i and g^{update} is update gradient from local.
Parameter Server:
Perform *Reduce* and gather all $g_{i,t-1}^{update}$
Averaging gradients $\bar{g}_{t-1} = \frac{1}{n}\sum_i g_{i,t-1}^{update}$
Perform *Broadcast* $\bar{g}_{i,t-1}$ to workers $i \in [N]$
On i-th worker
Update model parameter $x_t \leftarrow x_{t-1} - \gamma_t \bar{g}_{i,t-1}$
Compute local gradients $g_{i,t} = \nabla f(x_t; D_i)$
if Gradient quantization **then**
  $g_{i,t}^{update} \leftarrow \text{Quantization}(g_{i,t})$
else if Gradient sparsification **then**
  $g_{i,t}^{update} \leftarrow \text{Sparsification}(g_{i,t})$
else
  $g_{i,t}^{update} = g_{i,t}$
end if
Send $g_{i,t}^{update}$ to Parameter Server

More specifically, in the context of PS based distributed scheme, the standard distributed SGD algorithm can be expressed as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\gamma_t}{n}\sum_{t=1}^{n} g_t \tag{4}$$

Algorithm 1 illustrates a generalized communication efficient distributed algorithms.

III. Communication efficient algorithm: CE-SGD

In this section, we will introduce our communication efficient distributed training algorithm CE-SGD. We shed light on the analysis of communication reduction of different schemes. We also prove the convergence of CE-SGD on both convex and non-convex problems.

A. CE-SGD Algorithm

The main idea of the algorithm is to transmit a gradient with a probability according to its usage. For this purpose, we define a selection vector $V_{i,m,t}$ (definition 1) as a mask to determine whether a gradient will be transmitted. When an element of the vector is 1, the gradient will be transmitted, otherwise it will not. The sparse operator (definition 2) works on generating a selection vector. The element of the selection vector corresponding to a gradient will be more likely to be 1 if (1) previously sending the exact gradient will make the loss function decrease and (2) the gradient has been selected multiple times (stored as weighted coordinate map). In essence, the change of the loss function based on a gradient will give feedback to the worker to show whether the transmission of the gradient will contribute to reducing the loss function. The worker will transmit the gradient that will be more likely to reduce the loss function based on the observation in the previous rounds. Before moving on to our algorithm,

Algorithm 2 CE-SGD

Initialization Initialize model parameter x_i for all workers. Learning rate $\gamma_0 = 0.001$. Weighted Coordinate Map $M_0 = \mathbf{0}$. A random selection vector V_0.
Parameter Server:
Perform *Reduce* and gather all $g_{i,m-1,t-1}^{update}$
Averaging gradients $\bar{g}_{,m-1,t-1} = \frac{1}{n}\sum_i g_{i,m-1,t-1}^{update}$
Perform *Broadcast* $\bar{g}_{i,m-1,t-1}$ to workers
On i-th worker
Compute local loss $f_i(x_t, \mathcal{D}_i)$
for Model Layer $1, \cdots, L$ **do**
  Update $m-1 \in [L]$ layer parameter $x_{i,m-1,t} \leftarrow x_{i,m-1,t-1} - \gamma_t \bar{g}_{i,m-1,t-1} \odot V_{i,m-1,t-1}$
  Compute m-th layer gradient $g_{i,m,t}$ by backpropgation
  if $f(x_{t-1}, D_i) > f(x_t, D_i)$ **then**
    Retain selection vector $V_{i,m,t} \leftarrow V_{i,m,t-1}$
    $g_{i,m,t}^{update} \leftarrow g_{i,m,t} \odot V_{i,m,t}$
    Update weighted coordinate map $M_{i,m,t} = \mathcal{M}_w(V_{i,m,1\to t}) = \sum_{t=1}^{t} V_{i,m,t}$
  else
    Calculate selection vector $V_{i,m,t} \leftarrow \rho(M_{i,m,t-1})$
    $g_{i,m,t}^{update} \leftarrow g_{i,m,t} \odot V_{i,m,t}$
  end if
  Speculative Update: Randomly choose some rare-update coordinate to $g_{i,m,t}^{update}$
  Send $g_{i,m,t}^{update}$ to Parameter Server
end for

we introduce the selection vector, sparse operator and sum weighted coordinate map as following.

Definition 1. *(Selection Vector $V_{i,m,t}$) We define $V_{i,m,t} \in [0,1]^k$ as the selection vector of gradient $g_{i,m,t}$ where k is the dimension of gradient $g_{i,m,t}$.*

Selection vector $V_{i,m,t}$ of i-th worker's model m-th layer only select gradient $g_{i,m,t}^{(e)}$ if and only if $V_{i,m,t}^{(e)} = 1$. Selection vector is generated via applying a sparse operator ρ on a sum weighted map M_t which accumulates all previous selection vectors that have been used for each iteration.

Definition 2. *(Sparse operator) Given a non-negative defined vector $v_t \in R^k$, the j-th entries of sparse vector μ_t w.r.t vector v_t is defined as*

$$\mu_t^{(j)} \triangleq \begin{cases} 1, & \textit{with the probability } \rho^{(j)}(v_t) \\ 0, & \textit{otherwise} \end{cases} \tag{5}$$

where the sparse operator $\rho(v_t) \in [0,1]^k$. For weighted sparse operator, we define $\rho^{(j)}(v_t) = |v_t^{(j)}|/\|v_t\|_2$ for normalization purpose. For Bernoulli operator, the $\rho^{(j)}(v_t) =$ Bernoulli$(|v_t^{(j)}|/\|v_t\|_\infty)$.

We also define weighted coordinate mapping function $\mathcal{M}$ as

Definition 3. *Given selection vector $V_{i,m,t}$ defined above, coordinate mapping function $\mathcal{M}$ takes $V_{i,m,1}, V_{i,m,2}, \cdots V_{i,m,t}$ as input and return an accumulated map M_t where*

$$M_t = \mathcal{M}(V_{i,m,1}, V_{i,m,2}, \cdots, V_{i,m,t}) \tag{6}$$

In this paper, we use sum weighted gradient coordinate mapping function $\mathcal{M}_w$ where

$$M_t = \mathcal{M}_w(V_{i,m,1\to t}) = \sum_{t=1}^{T} V_{i,m,t} \tag{7}$$

At this point, we can present the synchronous distributed communication efficient algorithm in Algorithm 2. PS performs the sparse synchronization via the *Reduce* and then *Broadcast* averaging gradients to each worker (line 2-4). Each worker keeps an identical model copy and maintains a tuple of 1) layer-wise gradients $g_{i,m,t}$ computed in each iteration; 2) a sum weighted coordinate map $M_{i,m,t}$ that keeps tracking the coordinate-wise weight changes of layer-wise gradients and 3) a selection vector $V_{i,m,t}$ at current step t that marks which gradient coordinate can be used for PS synchronization. It is worth mentioning that we use the Bernoulli operator to generate sparse selection vector. That is our sparsification function is defined as $\rho^{(j)}(g_{i,m,t}) = \text{Bernoulli}(|g^{(j)}_{i,m,t}|/\|g_{i,m,t}\|_\infty)$.

On each worker, we keep the loss value of last update $f(x_{t-1})$ for current iteration t. We train the model on local dataset D_i and do backpropagation layer-wisely. For the gradient of m-th layer, we use loss feedback mechanism (line 8) and selection vector $V_{i,m,t}$ to decide which gradient entries will be upload to the PS for next step synchronization. When the learning model receive a positive feedback (for example $f(x_{t-1}) > f(x_t)$), we think the current gradient coordinate selection will reward the model and we increase the probability of that coordinate being selected in the next iteration by fine-tuning the weighted coordinate map (line 9-12). On the opposite, if negative feedback is received (line 13) such as $f(x_{t-1}) > f(x_t)$. It means the current gradient coordinate selection is not beneficial for the model. Therefore we are looking for new gradient coordinate selection. The CE-SGD then generates sparse gradient by applying sparse operator on weighted coordinate map (line 14-15).

In order to avoid some gradient coordinates never being updated and ensuring the unbiasedness, we randomly select coordinates that have a small weight value or the coordinate that has not been updates for long time (line 17). We call this process as speculative update. Speculative update can be performed in a periodical manner. Figure 2 explains how to generate sparse gradient according to sparse operator on coordinate mapping function $\mathcal{M}$.

B. Communication Reduction Analysis

To analyze how much worker-to-server traffic that CE-SGD can reduce, we make the following assumptions. For simplicity, we use a non-dropout and non-skipped deep neural network with L layers in total. We suppose it has all fully connected layers in the set FC and other layers in the set NFC such that $|FC| + |NFC| = L$. We define the dimension of its m-th layer as k_m. Let us assume the ratio of the total number

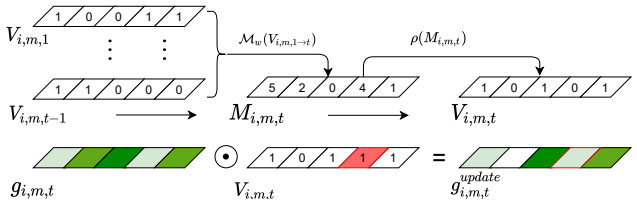


Fig. 2: This figure demonstrates how the selection vector $V_{i,m,t}$, sparse operator ρ and coordinate mapping function $\mathcal{M}$ work together. In order to update selection vector $V_{i,m,t}$ to $V_{i,m,t+1}$, we sum up all previous selection vector element-wise from t = 1 to t for example $M_{i,m,t} = \sum_{t=1}^{t} V_{i,m,t}$. Then we apply Bernoulli operator ρ to $M_{i,m,t}$ such that $V_{i,m,t} = \rho(M_{i,m,t-1})$. The update gradient $g^{update}_{i,m,t}$ is computed by Hadamard product of gradient $g_{i,m,t}$ and current selection vector $V_{i,m,t}$ shown as green colored coordinates. The coordinate in red box is selected because of the speculative update where we randomly select some coordinates for synchronization.

of non-fully-connected layer parameters to fully-connected layer parameters is $1/c < 1$. For gradient quantization, the quantization level is l. The layer-wise gradient drop ratio is $1 - r_{V_{m,t}}$ at iteration t. Suppose $r_{V_{m,t}} \sim \mathcal{N}(\mu, \sigma^2)$ over t and $\mu \in (0, 1)$. As a results, the bits that required by gradient quantization per communication round are:

$$\begin{aligned} &32* \sum_{m\in NFC} k_m + \sum_{m\in FC} k_m(\log_2(2*l+1)) + 32 \\ &= (\frac{32}{c} + (\log_2(2*l+1)) \sum_{m\in FC} k_m + 32 \end{aligned} \tag{8}$$

Meanwhile, for the CE-SGD:

$$32 * \sum_{m\in NFC\cup FC} k_m r_{U_{m,t}} = 32(\mu/c + 1) \sum_{m\in FC} k_m \tag{9}$$

When we use gradient quantization level $l = 4, 6, 8$ and FC/NFC ration $c < 1/\mu - \mu$, CE-SGD outperforms other algorithms.

C. Convergence of CE-SGD

In this section, we will analyze the convergence of CE-SGD on convex and non-convex optimization problems. The theoretical results are consistent with our experimental results. We present two theorems below, the proofs of which are omitted due to the page limit.

Theorem 1. *(Convex Case) Suppose the loss function $f(x)$ is convex and differentiable on R^d. We assume that the optimal set X^* of our objective function $f(x)$ is nonempty and bounded. $f(x)$ has bounded gradients $\mathbb{E}||\nabla f(x)|| \le G$. Let the learning sequence $\{\gamma_t\}$ be $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$ and $\sum_{t=1}^{\infty} \gamma_t = \infty$. Then the iterative CE-SGD shown in Algorithm 2 converges to a stationary point s.t. $f(x_t) - f(x^*) \le C_\gamma ||x_0 - x^*||$ where x_0*

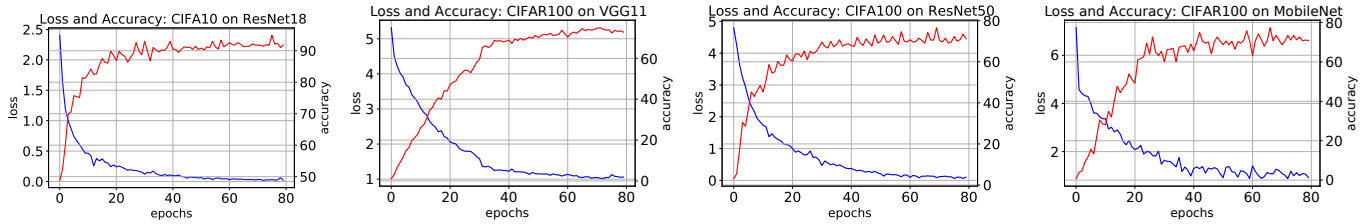


Fig. 3: Loss (in blue color) and Top-1 accuracy (in red color) of CIFAR10 on ResNet18 and CIFAR100 on VGG11, ResNet50 and MobileNet. In all experiments, we use Adam optimizer with initial learning rate 0.001. The min-batch size is 64 which is smaller than other experiments. This is due to the limited memory and computational resource on Raspberry Pi.

is random initial point and $C_\gamma = 2/\sum \gamma_t$. *We also conclude that*

$$f(x_t) - f(x^*) \leq \mathcal{R}_{x_0} ||\nabla_{U_{t,i}} f(x_t)|| \tag{10}$$

where $\mathcal{R}_{x_0} = \sup_{x_t}\{\max_{x \in X^*} ||x_t - x^*|| : f(x_t) \leq f(x_0)\}$

The following theorem shows the convergence of CE-SGD on the non-convex problems.

Theorem 2. *(Non-convex case) We assume that the optimal set* X^* *of our loss function* $f(x)$ *is nonempty and bounded.* $f(x_t)$ *has bounded gradients* $\mathbb{E}||\nabla f(x)|| \leq G$. *Let the learning sequence* $\{\gamma_t\}$ *is* $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$ *and* $\sum_{t=1}^{\infty} \gamma_t = \infty$. *And we define* $\phi_t \equiv E[f(x_t)|\mathcal{I}_t]$. *Algorithm 2 converges to a stationary point in expectation as:*

$$\phi_T \leq \phi_0 - \frac{1}{2}\sum_{t=0}^{T} \gamma_t \mathbb{E}||\nabla f(x_t)||^2 + \frac{1}{2}\sum_{t=0}^{T} \gamma_t \kappa_t \tag{11}$$

In the next section, we show our experimental results and analysis.

IV. Numerical Results

In this section, we validate our algorithm with experiments on various machine learning image classification and language tasks. Compared to the state-of-the-art efficient algorithms, CE-SGD outperforms in convergence speed, test accuracy and communication costs.

A. Experiment Setup

Environment In order to simulate the real world scenario, we conduct experiments on the distributed system that contains 4 Raspberry Pi 3 Model B as remote workers and a PC as central parameter server. We link Raspberry Pi with PS via standard wireless connection protocol. Raspberry Pi has one Camera Module port and one Audio jack which make it to be capable of any image classification or language recognition tasks.

Datasets and networks The datasets we use for image classification include: MNIST, FashionMNIST, CIFAR10, CIFAR100 and ImageNet (ILSVRC2013). The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. Fashion-MNIST consists of the same number of train samples and test samples as MNIST. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million 32x32 colour image dataset. ImageNet consists of two parts, training data and validation data. The training data contains 1000 categories and 1.2 million images and the validation and test data consists of 150,000 photographs, collected from flickr and other search engines, hand labeled with the presence or absence of 1000 object categories. The training models we are using include LeNet, AlexNet, ResNet, VGG, GoogleNet and MobileNet. LeNet contains 60k parameters and AlexNet contains 60M parameters with actual model size 233MB. ResNet (actual size 44.7 MB) and GoogleNet (also known as InceptionV3, 104MB) are typical very deep neural networks and they contain 12M and 23M parameters respectively.

TABLE I: Communication Cost for Large Scale Training

Method	Cost Per-round (MB)	Top-1	Top-5
BaseLine	66	77.55%	93.37%
GradientDrop	36.2	65.40%	89.12%
QSGD	24.6	71.48%	92.89%
CE-SGD	**21.1** (Avg.)	**75.13%**	**93.64%**

B. Communication reduction

We first evaluate the convergence of CE-SGD as shown in Figure 3. All four results are achieving the state-of-the-art accuracy. Further, we evaluate the communication cost of gradient transmission cost per-round as shown in Table I. We train CIFAR100 on GoogleNet. The raw size of GoogleNet is 104MB with all trainable parameter and hyper-parameters and the actual transmitted gradient size is 66MB. The highest drop ratio that gradient drop can achieve is 0.45 and the communication cost per-round is 36.2MB. If we choose some large drop ratio (≥ 0.45), the model accuracy will significantly drop and it occasionally does not even converge. For quantlized approach QSGD, theoretically, it can reduce the worker-to-server traffic by a factor of $32/log_2(8) \approx 10.3$. However, we are unable to achieve this amount of reduction in reality. Because, in gradient quantization schemes, communication

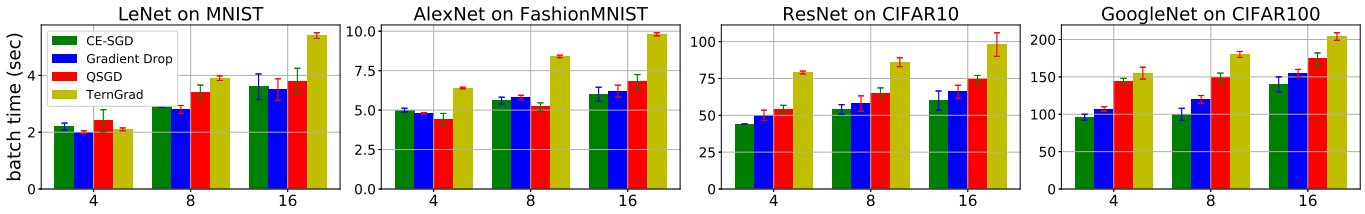


Fig. 4: The average batch time is defined as the cost of time of one round communication including the local training time, network transmission time and global aggregation time. When we increase the number of training workers, the network delay becomes a bottleneck of TernGrad because TernGrad uses a shared scalar [7].

Model	Cost/Actual Size (MB)	# Epochs	Top-1
MobileNet	**11**/14	120	76.12%
VGG11	**44**/89	80	76.41%
WideResNet	**27**/62	80	78.82%

TABLE II: Experiments on Respberry Pi 3 (CIFAR100)

reduction only benefits from quantization on the fully connected layers but not on convolution layers. The precision loss due to vector quantization and extra quantization variance will hurt convolution weights. CE-SGD, instead, applies adaptive sparsed gradient selection and lets the model decide what gradient is best for training. Unlike the QSGD, CE-SGD can apply gradient sparsification over all model layer including the convolution layer since we do not sacrifice gradient precision. In Table I, CE-SGD reduces the communication cost to 21.1MB per-round and achieves higher Top-1 and Top 5 accuracy (75.13% and 93.64% respectively). Further, we track the averaged cost for MobileNet, VGG11 and WideResNet shown in Table II. It shows that the CE-SGD can achieve up to 50% and 60% communication reduction on VGG 11 and WideResNet respectively.

In addition, we also conduct the experiments of distributed learning by using Google Cloud Service. We create separate VM instances (4, 8, 16) which has an NVIDIA K80 GPU with 24 GB of GDDR5 memory. We use NVIDIA Collective Communications Library (NCCL) as the communication backend to perform distributed multiple GPUs training. By default, NCCL backend will use the standard TCP-based network interface for communication. Compared with QSGD and TernGrad, CE-SGD will introduce extra computation. It is necessary to evaluate whether the extra computations could cause delay in distributed training process or not. As shown in Figure 4, the overhead extra computation of CE-SGD is negligible. When the complexity of DNNs is low and number of training workers is small, CE-SGD uses similar batch time like other methods. However, when the model grows bigger and more training workers involve in, the CE-SGD shows its advantage on batch time.

V. Conclusion

In this paper, we present the communication-efficient SGD algorithm to improve the learning efficiency for large-scale distributed optimization. The CE-SGD algorithm can effectively reduce the communication cost without observable learning accuracy loss by introducing the error feedback scheme. We analyze its convergence behavior from the theoretical perspective and demonstrate its advantage over state-of-the-art algorithms. Our experiments demonstrate the efficacy of the proposed algorithm.

Acknowledgements

We thank all reviewers for their helpful comments. This project was supported in part by US National Science Foundation grant CNS-1816399. This work was also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

References

[1] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems 24*. USA: Curran Associates, Inc., 2011, pp. 693–701.

[2] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *NIPS*. USA: NIPS, 2012.

[3] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. USA: Curran Associates, Inc., 2014, pp. 19–27.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[5] D. Alistarh, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: randomized quantization for communication-optimal stochastic gradient descent," *CoRR*, vol. abs/1610.02132, 2016.

[6] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns," in *Interspeech 2014*. USA: Interspeech 2014, 2014.

[7] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *CoRR*, vol. abs/1705.07878, 2017.

[8] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, vol. abs/, 2016.

[9] A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz, "SGD learns over-parameterized networks that provably generalize on linearly separable data," in *International Conference on Learning Representations*. USA: ICLR, 2018.

[10] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *INTERSPEECH*. USA: INTERSPEECH, 2015.
[11] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2020.