Communication Efficient SGD via Gradient Sampling with Bayes Prior

Liuyihan Song*, Kang Zhao*, Pan Pan, Yu Liu, Yingya Zhang, Yinghui Xu, Rong Jin
Machine Intelligence Technology Lab, Alibaba Group
{liuyihan.slyh, zhaokang.zk, panpan.pp, ly103369, yingya.zyy, renji.xyh, jinrong.jr}@alibaba-inc.com

Abstract

Gradient compression has been widely adopted in data-parallel distributed training of deep neural networks to reduce communication overhead. Some literatures have demonstrated that large gradients are more important than small ones because they contain more information, such as Top-k compressor. Other mainstream methods, like random-k compressor and gradient quantization, usually treat all gradients equally. Different from all of them, we regard large and small gradients selection as the exploitation and exploration of gradient information, respectively. And we find taking both of them into consideration is the key to boost the final accuracy. So, we propose a novel gradient compressor: Gradient Sampling with Bayes Prior in this paper. Specifically, we sample important/large gradients based on the global gradient distribution, which is periodically updated across multiple workers. Then we introduce Bayes Prior into distribution model to further explore the gradients. We prove the convergence of our method for smooth non-convex problems in the distributed system. Compared with methods that running after high compression ratio at the expense of accuracy, we pursue no loss of accuracy and the actual acceleration benefit in practice. Experimental comparisons on a variety of computer vision tasks (e.g. image classification and object detection) and backbones (ResNet, MobileNetV2, InceptionV3 and AlexNet) show that our approach outperforms the state-of-the-art techniques in terms of both speed and accuracy, with the limitation of 100$\times$ compression ratio.

1. Introduction

Recently, deep learning has achieved great success in many tasks, such as image classification [37, 45, 36], object detection [23, 32, 28], video understanding [46, 8, 47] and so on. By taking advantage of data-parallel distributed training equipped with more and more GPU resources, the expensive computational time-consuming of training deep

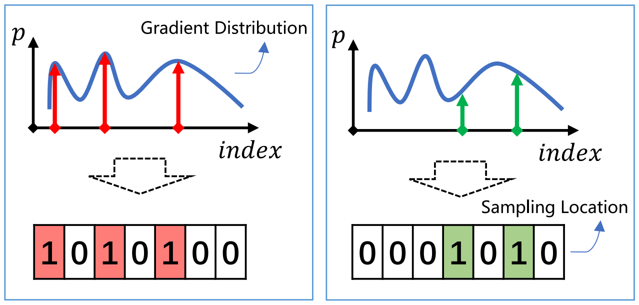

Figure 1. The core idea of our gradient compression method. The left subfigure illustrates that we sample important gradients based on gradient distribution. The right subfigure indicates that we also select some "trivial" ones as the exploration of gradient information.

neural networks have been dramatically reduced. However, the communication cost keeps rising with the increasing number of computing nodes. Even in All-Reduce architecture [31, 40], the transmission delay for communication is obvious. To handle this problem, gradient compression techniques [44, 22, 6, 14] have been put forward to reduce gradients transfer data size, by substituting partial gradients for full ones.

Although gradient compression has been well studied in [44, 22, 6], it is still a great challenge to maintain the accuracy at a high compression ratio. [44] combines error reset with partial synchronization to scale the compression ratio up to 1024$\times$ successfully. But the random sampling strategy adopted in partial synchronization overlooks the importance of large gradients. In [14], two adaptive gradient quantization schemes: ALQ and AMQ are proposed to improve the final accuracy. Unfortunately, it still suffers from some accuracy loss (more than 1.2% on ImageNet dataset), due to the limited representations of low-bit.

To better capture the gradient statistics, in this paper, we propose a new gradient compression method *Gradient Sampling with Bayes Prior* to further improve the accuracy. Compared with a very high compression ratio, we pursue no loss of accuracy and the actual acceleration benefit in practice. The major contributions of our work are outlined

*Equal Contribution.

as follows:

- Based on the global gradient distribution, we propose a novel gradient compression method called *Gradient Sampling* to efficiently capture the large gradients.
- We improve *Gradient Sampling* scheme with Bayes Prior to trade off the exploration and exploitation of gradients information, which boosts the final accuracy further.
- We prove the convergence bound of our proposed methods, and verify the convergence rate of our methods are the same as SGD under common assumptions.
- Experimental results on a variety of computer vision tasks and backbones show that our method is superior to the state-of-the-art techniques in terms of both speed and accuracy.

2. Related Works

Broadly, gradient compression methods can be roughly divided into two main categories: gradient sparsification and gradient quantization. We will introduce them respectively.

2.1. Gradient Sparsification

With only a subset of components of the original stochastic gradient selected, gradient sparsification produces sparse vectors. We often classify sparsification compressors into two major categories: fixed-dimension compressors and variable-dimension compressors. Fixed-dimension compressors mainly include two sparsification operators: Random-k and Top-k sparsifiers. Random-k sparsifiers [38, 12, 2] randomly sample k elements from the gradients for communication, it can not guarantee that large ones are selected. Top-k sparsifiers [6, 4, 27] needs to sort the gradients to select the largest k elements, resulting in extra computation overhead that can not be ignored in practice. Threshold-v [11] retains values whose magnitudes are larger than v and is a variable-dimension compressor. It is built on top of All-Gather architecture, which will incur extra index-value encoding. [42] proposes a variance-bounded coding strategy by readjusting the magnitude of randomly-dropping components in gradients. Yet unlike our work, no large-scale experiment has been conducted in that paper.

2.2. Gradient Quantization

It will quantize the gradients into low-precision values to reduce the communication overhead. 1-bit SGD [35] quantizes the gradients aggressively to one-bit per value for specific recurrent networks. EF-SIGNSGD [19] generalizes the 1-bit quantization with arbitrary compression operator and has the same rate of convergence as SGD. [3] proposes QSGD that can change the number of bits for quantization per iteration to balance communication bandwidth and convergence time. TernGrad [43] is a similar work to QSGD, which uses ternary gradients to accelerate distributed deep learning training. By observing the statistics of gradients during training process, [14] introduces two adaptive quantization schemes called ALQ and AMQ to improve the validation accuracy. Although gradient quantization can reduce 32-bits communication to the number of quantized bits, all of above methods suffer from more or less accuracy loss. Because the low-bit representation of gradients will lose more or less information, especially for large gradients, the quantization error will be larger.

3. Our Method

This section describes the formulation of our gradient compression framework. First, we introduce the motivation of our method. Then, we describe the *Gradient Sampling* compressor and its enhanced version combined with *Bayes Prior*. We prove the convergence of both algorithms under the most common assumptions. Finally, a complete pipeline of our compressor is given. To facilitate our discussion, some notations are given below.

Our aim is to solve the following minimization problem

$$\underset{\mathbf{x}}{min} f(\mathbf{x}) := \tfrac{1}{N}\textstyle\sum_{n=1}^{N} f_n(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the model parameters, $f_n : \mathbb{R}^d \mapsto \mathbb{R}$ is the loss function corresponding to each training sample n, with N total samples. The classic data-parallel stochastic gradient descent (SGD) can be formulated as:

$$\mathbf{g}_t = \triangledown f_n(\mathbf{x}_t),\ \mathbf{g}_t^* = Sync(\mathbf{g}_t),\ \mathbf{x}_{t+1} = \mathbf{x}_t - \eta\mathbf{g}_t^*. \tag{2}$$

At each iteration $t, t \in (0, T]$, $\mathbf{g}_t$ is local gradient calculated in each worker, $\mathbf{g}_t^*$ is global gradient after synchronization and η is the learning-rate.

3.1. Motivation

According to different gradient attentions, we divide the current mainstream gradient compression methods into two categories: i) **Treating large gradients more important than small ones.** The representative method is Top-k sparsifier. ii) **Treating all gradients equally,** such as Random-k sparsifier and Gradient Quantization schemes. None of them consider the two issues simultaneously.

Inspired by Reinforcement Learning [16, 5], sampling large gradients can be regarded as the exploitation of gradient information, which will maximize the reward of current training iteration. At the same time, we should take the exploration of gradient information (i.e., sampling small gradients) into account [7, 41]. Especially in distributed training system, the increasing number of GPUs lead to the decrease of training times and sampling rate, merely concentration on large gradients will reduce the communication of "trivial" gradients nearly to zero, which is harmful to the final accuracy. As a result, our goal is to balance the gradient exploitation and exploration with negligible computation overhead.

3.2. Gradient Sampling

3.2.1 Algorithm Description

At each iteration t, suppose we have access to global gradient $\nabla f(\mathbf{x}_t)$ (how to get global gradient will be shown in section 3.2.3). We define the sampling probability as follows:

$$p_{t,i} = \frac{\nabla f(\mathbf{x}_t)_i^2}{\|\nabla f(\mathbf{x}_t)\|^2}, \tag{3}$$

where $\nabla f(\mathbf{x}_t)_i$ $(0 \leqslant i < d)$ denotes i^{th} gradient element and $\|\nabla f(\mathbf{x}_t)\|^2 = \sum_i \nabla f(\mathbf{x}_t)_i^2$. It is clear that large element has large $p_{t,i}$.

Then, we draw a binary variable (0 or 1) from a Bernoulli distribution $B(p_{t,i})$. 1 means this element is sampled, 0 is otherwise. Bernoulli sampling can make the gradient with larger $p_{t,i}$ be selected for communication with higher probability.

In practice, in order to ensure that k elements in $\nabla f(\mathbf{x}_t)$ are sampled, we do the following transformation: $p_{t,i} = kp_{t,i}$ to make $\sum_i p_{t,i} = k$. After the transformation, $p_{t,i}$ could be larger than 1. To satisfy the condition of Bernoulli sampling $(0 \leq p_{t,i} \leqslant 1)$, we discard the part that exceeds 1, i.e., let $p_{t,i} = \min(p_{t,i}, 1)$. Then we increase the $p_{t,i}$ of unsampled ones, and repeat the sampling process in case the number of sampled elements is less than k. Algorithm 1 summarizes the iterative procedure.

According to Algorithm 1, the ultimate sampling probability can be represented as:

$$p_{t,i} = \begin{cases} \kappa \nabla f(\mathbf{x}_t)_i^2 / \|\nabla f(\mathbf{x}_t)\|^2, & \frac{\nabla f(\mathbf{x}_t)_i^2}{\|\nabla f(\mathbf{x}_t)\|^2} < \theta \\ 1, & \frac{\nabla f(\mathbf{x}_t)_i^2}{\|\nabla f(\mathbf{x}_t)\|^2} \geqslant \theta \end{cases}, \tag{4}$$

where κ is a normalization constant, and θ is a dynamic threshold controlling whether gradient element is fully sampled (both κ and θ are adaptively computed in Algorithm 1).

In Eq. (4), the gradients larger than $\theta\|\nabla f(\mathbf{x}_t)\|^2$ will be chosen for synchronization, which can be seen as a variant of Top-k compressor. It is noted that our Gradient Sampling is different from Top-k compressor from two aspects: i) We may sample some relatively large but not in Top-k gradients, i.e., the result of Algorithm 1 does not completely coincide with that of Top-k compressor, which is consistent with the nature of exploitation. ii) The computational time of Top-k compressor is O($d * log(k)$). $log(k)$ can not be ignored when the backbone is very huge. We don't have to sort the gradients to get the exact Top-k list, avoiding heavy computational overhead.

Algorithm 1 Iterative Distribution Normalization.

```
Input: The sampling probability $p^{samp} \in \mathbb{R}^d$, the number of sampling elements $k, k \in (0, d]$, maximum iteration $T_{max}$.
Output: normalized sampling probability $p^{norm}$ and normalized constant $\kappa$.
1: $\forall i, p_{0,i}^{norm} = \frac{k(p_{0,i}^{samp})^2}{\sum_i (p_{0,i}^{samp})^2}$
2: $\kappa_0 = k_0 = k$
3: for $j \leftarrow 1$ to $T_{max}$ do
4:     // Collect the set of fully-sampling gradients
5:     $S_j = \{\forall i, p_{j-1,i}^{norm} > 1\}$
6:     $l = |S_j|$ // the size of $S_j$
7:     $\forall i, p_{j,i}^{norm} = min(p_{j-1,i}^{norm}, 1)$
8:     if $l == 0 \,||\, k_{j-1} \leq 0$ then
9:         break
10:    else
11:        $s = \frac{k_{j-1} - l}{\sum_{i \notin S_j} p_{j,i}^{norm}}$
12:        $\forall i, p_{j,i}^{norm} = p_{j,i}^{norm} \times s$
13:        $\kappa_j = \kappa_{j-1} \times s$
14:        $k_j = k_{j-1} - l$
15:    end if
16: end for
17: Return $p_j^{norm}, \kappa_j$.
```

3.2.2 Theoretical Guarantees

In this section, we present the analysis for convergence rate of Gradient Sampling. As usual, we introduce some widely adopted assumptions.

A1. $f(\mathbf{x}_t)$ is bounded below: $\exists f^\star, f(\mathbf{x}_t) \geq f^\star$.

A2. $f(\mathbf{x}_t)$ is L-Lipschitz continuous: $f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2$.

A3. $\nabla f(\mathbf{x}_t)$ has bounded variance: $\|\nabla f(\mathbf{x}_t)\|^2 \leq \sigma^2$.

A4. Gradients from different workers are independent and identically distributed.

Based on the assumptions above, we can prove that Algorithm 1 has the same asymptotic convergence rate as vanilla SGD. Here, we give the framework and key lemmas in our analysis, and the missing proves are presented in Appendix A.

We start from applying Eq. (4) in **A2**, which makes the expected loss function satisfy

$$\mathbb{E}[f(\mathbf{x}_{t+1})] \leq \mathbb{E}[f(\mathbf{x}_t)] - \eta \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] + \eta \underbrace{\mathbb{E}[\langle \nabla f(\mathbf{x}_t), \mathbf{e}_t \rangle]}_{V1} + \frac{\eta^2 L}{2} \underbrace{\mathbb{E}\left[\|\mathbf{g}_t\|^2\right]}_{V2}. \tag{5}$$

where $g_{t,i} = p_{t,i} \nabla f(\mathbf{x}_t)_i, \mathbf{e}_t = \nabla f(\mathbf{x}_t) - \mathbf{g}_t$. According to sampling probabilty in Eq. (4), we can divide the gradients into two parts: i) fully sampled gradients, which satisfied $\forall i \in S_f, \nabla f(\mathbf{x}_t)_i^2 \geqslant \theta\|\nabla f(\mathbf{x}_t)\|^2$; ii) partially sampled gradients of the remaining. We use l to denote the number of fully sampled gradients. By independently treating these two parts of gradients, we can explicitly derive the expectation of variance terms ($V1$ and $V2$) in Eq. (5) as follows.

$$V1 = \sum_{i \notin S_f} \mathbb{E}\left[\left(1 - \frac{\kappa \nabla f(\mathbf{x}_t)_i^2}{\|\nabla f(\mathbf{x}_t)\|^2}\right)\nabla f(\mathbf{x}_t)_i^2\right], \quad (6)$$

$$V2 = \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] - V1. \quad (7)$$

To further bound $V1$, we denote the contribution of fully sampled gradients to $\|\nabla f(\mathbf{x}_t)\|^2$ as

$$\sum_{i \in S_f} \nabla f(\mathbf{x}_t)_i^2 = \epsilon\|\nabla f(\mathbf{x}_t)\|^2, \quad (8)$$

where $\epsilon \geq l/d$. Then we can guarantee the following upper bound of $V1$

$$V1 \leq (1-\epsilon)\left[1 - \frac{\kappa}{d-l}(1-\epsilon)\right]\mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right]. \quad (9)$$

Obviously, under 100% sampling ratio, ϵ is equal to 1 which makes $V1$ zero, and thus $V2$ becomes exactly the same as the variance term (i.e., $\mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right]$) in vanilla SGD. Now we are at the position to give the convergence rate of gradient sampling method by combining Eq.(7), Eq.(9) and Eq. (5), which becomes

$$\min_t\left\{\frac{1}{T}\mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right]\right\} \leq \frac{\mathbb{E}\left[f(\mathbf{x}_0)\right] - f^\star}{T} + \underbrace{\left(1 - \frac{\eta L}{2}\right)(1-\epsilon)\left(1 - \frac{\kappa(1-\epsilon)}{d-l}\right)\sigma^2}_{V3} + \underbrace{\frac{\eta L}{2}\sigma^2}_{V4}. \quad (10)$$

In Eq. (10), $V4$ is identical to the variance term in vanilla SGD, and by making $\eta = \frac{1}{T}$, we achieve well-known $O\left(\frac{1}{T}\right)$ convergence rate. On the other hand, the extra variance term caused by gradient sampling is $V3$. In the worst case, i.e., gradients are randomly distributed (being impossible in real life), ϵ takes its maximum, which maximizes $V3$ and degrades the gradient sampling to a random sampling. By choosing $\epsilon \geq 1 - \frac{d-l}{2\kappa}\left(1 - \sqrt{1 - \frac{4\kappa}{d-l}\frac{\eta L}{2-\eta L}}\right)$, we can ensure $V3 \leq V4$, and thus gradient sampling achieves the same $O\left(\frac{1}{T}\right)$ convergence rate as vanilla SGD. More importantly, by choosing $\eta L = 1$, we can guarantee $V3 \leq V4$ for any ϵ.

3.2.3 Gradient Distribution Estimation

How to get the global gradient is a key issue in Gradient Sampling. We can use All-Reduce to obtain the global gradient in each iteration, but it goes against the intention of gradient compression (avoid full gradients communication). If we utilize local gradients to execute the Gradient Sampling, the index-value of sampled gradients are needed because the local gradients of each worker are different. On the other hand, local gradients are not as accurate as global gradients, especially the number of workers is very huge.

Considering the network training is a continuous process, we can assume the gradient distribution changes little in a short training interval τ, i.e., $\nabla f(\mathbf{x}_{t_1}) \approx \nabla f(\mathbf{x}_{t_2})$, when $|t_1 - t_2| < \tau$. So, we use the historical gradient distribution $\nabla f(\mathbf{x}_{t_0})$ as the sampling distribution for the subsequent τ training steps:

$$\forall t_0 < t < t_0 + \tau, \nabla f(\mathbf{x}_t) = \nabla f(\mathbf{x}_{t_0}). \quad (11)$$

$\nabla f(\mathbf{x}_{t_0})$ can be obtained by one All-Reduce communication. In the implementation of Gradient Sampling, τ is set as large as 100, so the communication overhead of getting $\nabla f(\mathbf{x}_{t_0})$ can be limited.

3.3. Gradient Sampling with Bayes Prior

3.3.1 Algorithm Description

To coordinate the Gradient Sampling, we employ the Bayes Prior to reach the trade-off between exploration and exploitation. First, we define the following prior probability:

$$p_{t,i}(\Theta) = \prod_t \Theta_i^t, \quad (12)$$

$$\Theta_i^t = \begin{cases} \alpha, & i \in S_{t-1} \\ 1, & i \notin S_{t-1}, \end{cases} \quad (13)$$

where S_{t-1} is the set of sampling coordinates in the previous $t-1$ training iterations, and $\alpha \in (0,1]$ represents the probability that the sampled coordinate will be resampled. If one element is not selected during the previous iterations, i.e., $i \notin S_{t-1}$, we set α to one, thus no prior information is introduced. Otherwise, α will reduce the probability of the element being sampled in previous steps. For example, if we set $\alpha = 0.8$, the prior probability of a coordinate being sampled three times in succession will reduce quickly to $0.8^3 \approx 0.5$. In each training iteration, $p_{t,i}$ undergoes L1 normalization after it is updated by Eq. (12).

Based on Eq. (12), we adopt Bayes' theorem [18] to express the posterior sampling probability as:

$$p_{t,i}(\Theta|G) = \frac{p_{t,i}(G|\Theta)p_{t,i}(\Theta)}{p_{t,i}(G)} \propto p_{t,i}(G|\Theta)p_{t,i}(\Theta), \quad (14)$$

where $p_{t,i}(G|\Theta)$ comes from Gradient Sampling as detailed in Eq. (4). We take Eq. (14) as sampling probability and still resort to Algorithm 1.

After introducing the Bayes Prior, we can sample all coordinates in a shorter training period. To verify this, we apply Gradient Sampling (**GS**) and Gradient Sampling with Bayes Prior (**GSB**) to gradients sampling of conv6 in

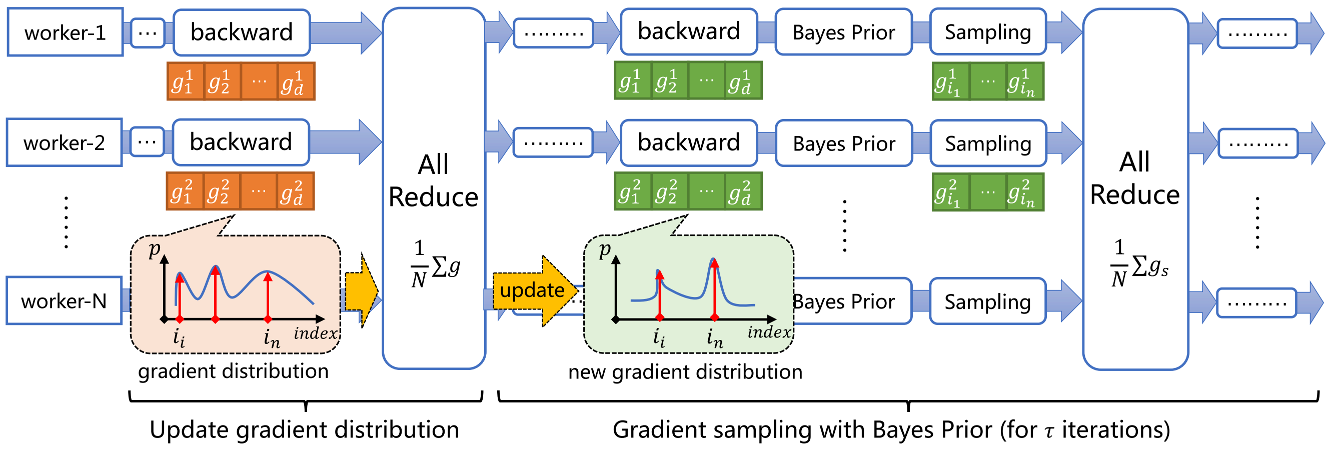
worker-1
worker-2
worker-N
backward
g_1^1 g_2^1 ··· g_d^1
g_1^2 g_2^2 ··· g_d^2
All Reduce
$\frac{1}{N}\Sigma g$
update
Bayes Prior
Sampling
$g_{i_1}^1$ ··· $g_{i_n}^1$
$g_{i_1}^2$ ··· $g_{i_n}^2$
$\frac{1}{N}\Sigma g_s$
p
index
gradient distribution
new gradient distribution
Update gradient distribution
Gradient sampling with Bayes Prior (for τ iterations)

Figure 2. Main workflow of Gradient Sampling with Bayes Prior. We use one All-Reduce to synchronize full gradients and update the gradient distribution. Then in the following τ iterations, we only communicate the sampled gradients based on the gradient distribution with Bayes Prior scheme.

ResNet-50. Under a high sampling ratio, e.g., $\phi = 0.1$, GS takes $200/\phi$ steps to sample all coordinates, while GSB only needs $2/\phi$ steps. As for a low sampling ratio, e.g., $\phi = 0.01$, GSB can make it within $5/\phi$ steps, whereas GS is almost impossible to sample all coordinates within limited steps. Overall, Bayes Prior can explore all gradient coordinates significantly faster.

3.3.2 Theoretical Guarantees

As discussed above, GSB aims to realize denser sampling for the larger gradient components (**Aim**$_1$), while guarantee a limited sampling times for the smaller gradients (**Aim**$_2$). However, due to the coupling of historical information in Bayes Prior, it is difficult to directly give a theoretical analysis. To overcome this problem, we design an alternative method which has similar sampling results as GSB when there are enough sampling steps. The key idea is to combine **T**op-**k** method (**Aim**$_1$) with **C**yclic **C**oordinate **D**escent sampling (**Aim**$_2$), and thus we name it as **TK-CCD**.

In TK-CCD, we first divide the entire optimization process into M stages, with each stage comprised of $Q = T/M$ trials. At the beginning of each stage m, we first choose the j components ($j \leq k$) with the largest magnitude of the global gradients, denoted by S_m. In addition, for the remaining components in the gradients, denoted by $\bar{S}_m$, we randomly divide it into Q groups, denoted by $\{U\} = U_0, U_1, ..., U_{Q-1}$, with each group comprising of $(d-j)/Q$ gradient components. Then, in the following Q trials, we sequentially choose one set from $\{U\}$, and the sampling gradient set becomes $S_m^q = S_m \cup U_q$. Using the same assumptions (**A1** to **A4**), we can bound the expectation of loss function as

$$\mathbb{E}\left[f\left(\mathbf{x}_m^{q+1}\right)\right] \leq \mathbb{E}\left[f\left(\mathbf{x}_m^q\right)\right] - \frac{\eta}{2}\mathbb{E}\left[\left\|\nabla f\left(\mathbf{x}_m^q\right)\right\|^2\right] + \frac{\eta L_s}{2}\mathbb{E}\left[\left\|\mathbf{x}_m^{q+1}|_{\bar{S}_m} - \mathbf{x}_m^q|_{\bar{S}_m}\right\|^2\right] + \frac{1-\eta L}{2\eta}\mathbb{E}\left[\left\|\mathbf{x}_m^{q+1} - \mathbf{x}_m^q\right\|^2\right], \quad (15)$$

where L_s is the Lipschitz constant for a single gradient component. We then choose ρ to make

$$\mathbb{E}\left[\left\|\mathbf{x}_m^{q+1}|_{\bar{S}_m} - \mathbf{x}_m^q|_{\bar{S}_m}\right\|^2\right] \leq \rho\mathbb{E}\left[\left\|\mathbf{x}_m^{q+1} - \mathbf{x}_m^q\right\|^2\right]. \quad (16)$$

By further having $2\eta^2\rho L_s \leq 1$, it is easily to show that Eq. (15) leads to the same $O\left(\frac{1}{T}\right)$ convergence rate as vanilla SGD under $\eta = \frac{1}{T}$.

Intuitively, GSB is a better implementation of TK-CCD, because: i) for gradients in $\bar{S}_m$, the TK-CCD does not consider the magnitude contribution; ii) GSB dynamically calculate the hyper-parameter j, which is required to be manually preset in TK-CCD; iii) utilization of cyclic coordinate descent limits the sampling ratio to be about $1/Q$, which is not low enough in practice. Indeed, we notice that GSB is an adaptive version of TK-CCD. Thus, by means of theoretical analysis in TK-CCD, we guarantee that GSB also has the same $O\left(\frac{1}{T}\right)$ convergence rate as vanilla SGD.

3.4. Efficient Implementation

We summarize the complete dataflow of applying our GSB for efficient communication in Figure 2. In this method, there are two types of synchronization, i.e., one for gradient distribution update (**Sync**$_1$), and the other for actual gradient communication (**Sync**$_2$). The sampling ratio of **Sync**$_1$ is determined by synchronization period (τ), being $1/\tau$, and that of **Sync**$_2$ is exactly k/d. Thus, the overall sampling ratio becomes $1/\tau + k/d$.

It must be noted that both **Sync**$_1$ and **Sync**$_2$ can be efficiently implemented by All-Reduce operation in any mainstream frameworks, e.g., Pytorch [30], TensorFlow [1], Caffe [17]. Moreover, the computationally intensive procedure of our method mainly comes from iterative distribution normalization in **Sync**$_2$. In practice, we limit the maximum iteration steps to a very low value (i.e., 5), and thus avoid extra computational overhead. The overall pipeline is presented in Appendix B.

4. Experiments

In this section, we compare our proposed method with many other state-of-the-art methods on extensive image classification and object detection tasks. In specific, the compared methods include DGC [26], dist-EF-BlockSGDM (EB-SGD) [48], QSparse-local-SGD (QL-SGD) [6], and CSER [44]. We briefly summarize the compared methods as follows.

DGC. Adopts a Top-k sparsifier to compress the gradients and employs momentum correction and local gradient clipping on top of the sparsifier to maintain model performance.
EB-SGD. Introduces a blockwise compressor by partitioning the gradient into blocks so that each block could be compressed and transmitted in 1-bit format with a scaling factor.
QL-SGD. Includes both quantization and sparsification in compressor, incorporates error-feedback to correct the residual errors, and also uses the infrequent synchronization method as in local-SGD [22] to decrease the overall number of communication rounds.
CSER. Introduces error reset to achieve better convergence when aggressive compressors are used, and adds a second compressor to partially synchronize the gradients on local models in each iteration.

To implement these methods we followed the open-source codes which are provided by these papers: DGC[*], EB-SGD[†], QL-SGD[‡], and CSER[§]. In order to test training speed in practice, we conduct all of the experiments on the GPU cluster with the following configurations: Up to 32 machines, and each machine is equipped with 2 NVIDIA Tesla P100 (16 GB), which are interconnected with the PICe bus. For network connectivity, each machine uses a 10-Gbps Ethernet card for message communication. The deep learning framework PyTorch [30] is used to implement the communication-efficient methods and we use Gloo [¶] as communication backend.

[*]https://github.com/synxlin/deep-gradient-compression
[†]https://github.com/ZiyueHuang/dist-ef-sgdm/
[‡]https://github.com/karakusc/horovod/tree/qsparselocal
[§]https://github.com/xcgoner/NeurIPS2020_CSER
[¶]https://github.com/facebookincubator/gloo

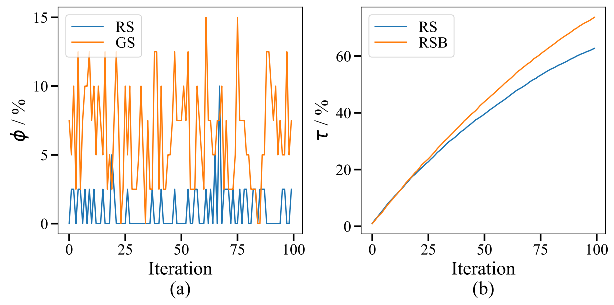

Figure 3. The evolution curves of ϕ/% (largest elements in the sampled gradients) and τ /% (gradients visited more than once) for a convolution layer in ResNet-50.

The overall compression ratio is chosen as 1% for all methods except EB-SGD and CSER. In our method, we sample 1% gradients for communication and update global gradient every 100 iterations (1%). They are transmitted using FP16 format resulting in 1% compression ratio in total. As for EB-SGD, due to the utilization of a 1-bit compressor, we set the compression ratio as its minimum, being $1/32$. The CSER use default recommendation setting to make overall $1/96$ compression ratio. In DGC, the sampling rate is directly set to 1%. In QL-SGD, the selected 8% largest gradients are quantized to 8-bit, and the communication interval are adjusted to match target ratio.

All experimental settings are consistent with the vanilla SGD to avoid extra hyperparameters tuning. Note that a multistep learning rate scheduler is used for most of the models, despite MobileNetV2 uses cosine scheduler [29] as widely adopted. We set batch size to 2048 for each experiment if there is no special explanation.

4.1. Ablation Study

We conduct the ablation study on ImageNet [10] with popular ResNet-50. The baseline is chosen as random sampling (RS), which serves as no additional knowledge of sampling.
Gradient Sampling. To verify the effectiveness of GS, we testify the ability to sample the most important gradients. Figure 3(a) shows the evolution of the proportion of the largest elements ϕ in the sampling elements within a short time period (i.e., 100 training iterations). As expected, ϕ_{RS} is only 1% on average, since gradients are impossible to be uniformly distributed. More importantly, we can find that ϕ_{GS} is about 8 times higher than that of ϕ_{RS} during the examining period, which supports GS can more effectively sample for the largest elements.
Bayes Prior. Our second experiment combines random sampling with Bayes prior (RSB) to get rid of GS. We compare the visited gradients τ in 100 continuous iterations using RS and using RSB. Figure 3(b) clearly shows that τ_{RSB}

Method	RS	Ours$_{\mathrm{GS}}$	Ours$_{\mathrm{RSB}}$	Ours
Acc / %	75.65	76.13	75.97	**76.41**

Table 1. Ablation study on different sampling strategies.

α	0.6	0.7	0.8	0.9	0.95	0.99
Acc.	76.27	76.33	76.41	76.41	76.39	76.35

Table 2. Ablation study on different sampling strategies.

are increasingly larger than τ_{RS} as iteration goes, which indicates that the addition of Bayes prior strongly boost the efficiency of exploration.

Results. Table 1 summarizes the final accuracy of different sampling strategies. Ours$_{\mathrm{GS}}$ and Ours$_{\mathrm{RSB}}$ obtain 0.48% and 0.32% accuracy improvement over Ours$_{\mathrm{RS}}$ respectively, showing again these two strategies are effective. Not surprisingly, our proposed framework Ours has the highest accuracy improvement, i.e., 0.76%, which exceeds both Ours$_{\mathrm{GS}}$ and Ours$_{\mathrm{RSB}}$. This demonstrates our method could achieve a good balance between exploitation and exploration of gradients.

To verify the stability of Bayes prior, we further examine the sensitivity of prior probability α. As Table 2 lists, the accuracy are similar when α is within $(0.8, 0.95)$. This is not surprising since too large α means no prior information while too small α means prior information are too strong to diminish the effect of GS. In the following text, we set $\alpha = 0.9$ for all examined experiments.

4.2. Image Classification

We show the classfication performance on CIFAR-10 [20] and ImageNet [10] datasets, with various backbones such as AlexNet [21], ResNet [15], InceptionV3 [39], and MobileNetV2 [34]. The results of distributed SGD optimization are presented as the baseline.

In the experiments of CIFAR-10, Table 3 shows that our performance is as accurate as SGD for all examined models. Especially, we surpass all other methods in ResNet-20 and InceptionV3. Surprisingly, DGC has more than 1% accuracy drop in MobileNetV2. This may be attributed to that depth-wise convolution requires sufficient gradient updating for each channel to stimulate the corresponding neurons, while only the largest gradients are transmitted in DGC.

Method	ResNet-20	InceptionV3	MobileNetV2
SGD	91.66	95.20	94.51
DGC[26]	91.13	94.67	93.44
EB-SGD [48]	91.31	94.46	94.24
QL-SGD[6]	91.50	95.03	**94.72**
CSER[44]	91.48	94.49	94.63
Ours	**91.80**	**95.13**	94.57

Table 3. The Top-1 accuracy on CIFAR-10 dataset.

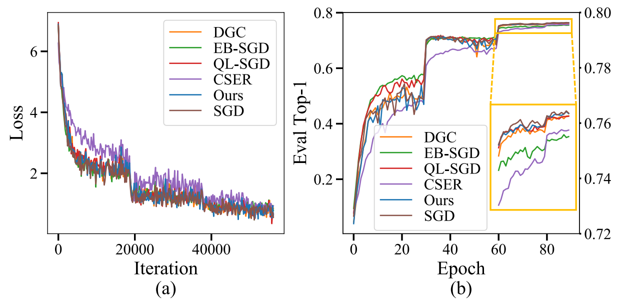

Figure 4. Comparison of different methods of training ResNet-50 on ImageNet. (a) Loss function on training dataset. (b) Top-1 accuracy on evaluation dataset.

Table 4 shows the overall Top-1 accuracy using different backbones with various methods on ImageNet. Our method has achieved almost negligible accuracy degradation ($\leq 0.2\%$) compared with distributed SGD. It is noted that our method has the best performance in most of the large models, i.e., ResNet-50, ResNet-101, InceptionV3, and MobileNetV2. This gives a direct evidence to the importance of considering both exploitation and exploration in compressing gradients. Among these compared communication-efficient methods, EB-SGD and CSER have a bit accuracy drop, because the former utilizes aggressive 1-bit compressor while the random sampler in the latter does not consider any magnitude contribution. The methods DGC and QL-SGD both get relatively good results on many backbones, but they all suffer from significant computational overhead due to the utilization of Top-k operation.

Taking ResNet-50 as an example, we demonstrate the evolution of training loss and validating results in Figure 4. We observe that the convergence rate of our method is nearly consistent with SGD, which empirically proves our theoretical derivation in section 3.3.2. The CSER has the slowest convergence rate, since it cause additional difficulty to train the network by separately synchronizing different parts of gradient information.

4.3. Object Detection

We now further test our method on popular detection networks, being Faster-RCNN [33] and RetinaNet [24], on COCO [25] and PASCAL VOC [13] datasets. We adopt the open-source MMdetection [9] framework with the default settings and take ResNet-50 as backbone. Here, the evaluation metric is chosen as mean Average Precision (mAP).

As listed in Table 5, our method achieves the best performance (less than 0.3%) for both Faster-RCNN (two-stage) and RetinaNet (one-stage) on large-scale COCO datasets. We note that DGC, QL-SGD, and our method surpass EB-SGD and CSER for a large margin ($\geq 0.5\%$), which indi-

Method	ResNet-18	ResNet-34	ResNet-50	ResNet-101	InceptionV3	MobileNetV2	AlexNet
SGD	70.22	73.46	76.44	77.59	77.32	71.86	58.15
DGC[26]	**70.16**	**73.40**	76.28	77.26	76.91	71.64	57.80
QL-SGD [6]	70.04	73.25	76.24	77.38	76.89	71.82	**58.14**
EB-SGD [48]	69.69	72.75	75.58	76.99	76.55	71.89	55.05
CSER[44]	69.98	72.85	75.76	76.86	76.60	71.21	57.53
Ours	70.13	73.38	**76.41**	**77.57**	**77.13**	**71.96**	58.10

Table 4. The Top-1 accuracy on ImageNet dataset.

Dataset	COCO		PASCAL-VOC	
Backbone	Retina -Net	Faster -RCNN	Retina -Net	Faster -RCNN
SGD	36.4	37.5	77.3	79.5
DGC[26]	35.7	37.2	77.1	**79.2**
QL-SGD[6]	35.9	37.1	77.0	78.9
EB-SGD[48]	34.8	36.5	75.0	77.2
CSER [44]	35.3	36.6	76.6	78.6
Ours	**36.1**	**37.4**	**77.3**	79.0

Table 5. Detection Results on COCO and PASCAL VOC Datasets.

cates the compressor based on random sampling will easily miss transmission for gradients of the most important anchors. On the small-scale PASCAL-VOC dataset, though DGC has slightly better performance than our method in Faster-RCNN, our method is still recommended due to less computational overhead.

4.4. Acceleration Results

The capability of gradient compression is to speed up training. For larger models, the communication overhead becomes a major factor. Such models are expected to benefit greatly with communication efficient methods. So we stand to evaluate the actual performance in different methods.

For the reason that the method as QL-SGD adopts infrequent synchronization, and each method has its own compressing operation which introduces different computation overhead, so we use the training throughput and wall-clock training time to measure the training speed for each one. Here, we use AlexNet which is quite intensive in terms of communication.

In Figure 5(a), we compare our proposed method with others using system throughput which describes the speed of processing training samples. It is clear that our proposed method is superior to other methods except for CSER. However, in Section 4.2 we observed that CSER might lose accuracy, so it may not be the first choice for precision sensitive tasks. As the number of workers participating in communication increases, the methods DGC, QL-SGD, and EB-SGD suffer a great loss of training speed, because they must

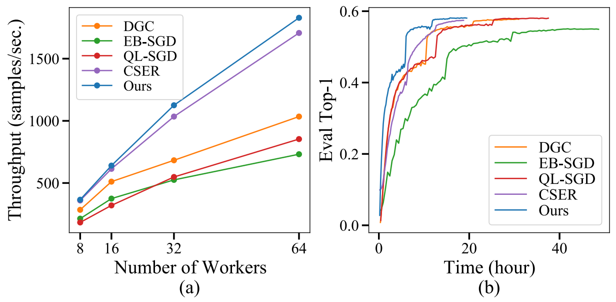

Figure 5. Comparison of training speed using AlexNet on ImageNet. (a) Throughput: Processing samples per second. (b) Evolution of Top-1 accuracy on validation dataset with respect to wall-clock time.

adopt sparse synchronization which is slow and inefficient. The other reason is they use such compress operations as Top-k or quantization which always introduces extra computation overhead while leading to a larger variance in latency of synchronization.

Figure 5(b) shows the wall-clock time of training. To reach comparable accuracy (less than 0.5% accuracy loss, CSER is not satisfied), our method outperforms other methods by more than 76%, which is a significant saving for large-scale training. Moreover, we achieve 2.5× improvement on ImageNet dataset in our training cluster. More acceleration experiments will be shown in Appendix C.

5. Conclusion

In this work, we propose a novel gradient compression method to balance the exploration and exploitation of gradient information. In particular, we employ Gradient Sampling to efficiently capture the large gradients based on the periodic updated global gradient distribution. Then Bayes Prior is introduced into the distribution model to boost the final accuracy further. We theoretically and empirically prove the convergence of our method. Our approach has successfully trained on a variety of computer vision tasks and networks with negligible accuracy degradation, which sets a new state-of-the-art to the community.

References

[1] 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 *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. 6

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 440–445, 2017. 2

[3] 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 (NeurIPS)*, pages 1709–1720, 2017. 2

[4] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5973–5983, 2018. 2

[5] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002. 2

[6] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14695–14706, 2019. 1, 2, 6, 7, 8

[7] Djallel Bouneffouf. Exponentiated gradient exploration for active learning. *Computers*, 5(1):1, 2016. 2

[8] Biagio Brattoli, Joseph Tighe, Fedor Zhdanov, Pietro Perona, and Krzysztof Chalupka. Rethinking zero-shot video classification: End-to-end training for realistic applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4613–4623, 2020. 1

[9] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 6, 7

[11] Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 3817–3824, 2020. 2

[12] Melih Elibol, Lihua Lei, and Michael I Jordan. Variance reduction with sparse gradients. In *International Conference on Learning Representations*, 2019. 2

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 7

[14] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel sgd. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 1, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 7

[16] Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687, 2002. 2

[17] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014. 6

[18] James Joyce. Bayes' theorem. 2003. 4

[19] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning (ICML)*, pages 3252–3261, 2019. 2

[20] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 7

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012. 7

[22] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 6

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2980–2988, 2017. 7

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 7

[26] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication

bandwidth for distributed training. In *International Conference on Learning Representations (ICLR)*, 2018. 6, 7, 8

[27] 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 Learning Representations (ICLR)*, 2017. 2

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems (NeurIPS)*, pages 8026–8037, 2019. 6

[31] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009. 1

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015. 7

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 7

[35] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Conference of the International Speech Communication Association (InterSpeech)*, 2014. 2

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[37] Liuyihan Song, Pan Pan, Kang Zhao, Hao Yang, Yiming Chen, Yingya Zhang, Yinghui Xu, and Rong Jin. Large-scale training system for 100-million classification at alibaba. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2909–2930, 2020. 1

[38] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4447–4458, 2018. 2

[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 7

[40] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005. 1

[41] Huazheng Wang, Sonwoo Kim, Eric McCord-Snook, Qingyun Wu, and Hongning Wang. Variance reduction in gradient exploration for online learning to rank. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 835–844, 2019. 2

[42] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1299–1309, 2018. 2

[43] 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 (NeurIPS)*, pages 1509–1519, 2017. 2

[44] Cong Xie, Shuai Zheng, Oluwasanmi O Koyejo, Indranil Gupta, Mu Li, and Haibin Lin. Cser: Communication-efficient sgd with error reset. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 1, 6, 7, 8

[45] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. 1

[46] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018. 1

[47] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3083–3092, 2020. 1

[48] Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum sgd with error-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11450–11460, 2019. 6, 7, 8