UNDERSTANDING TOP-K SPARSIFICATION IN DISTRIBUTED DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Distributed stochastic gradient descent (SGD) algorithms are widely deployed in training large-scale deep learning models, while the communication overhead among workers becomes the new system bottleneck. Recently proposed gradient sparsification techniques, especially Top-k sparsification with error compensation (TopK-SGD), can significantly reduce the communication traffic without obvious impact on the model accuracy. Some theoretical studies have been carried out to analyze the convergence property of TopK-SGD. However, existing studies do not dive into the details of Top-k operator in gradient sparsification and use relaxed bounds (e.g., exact bound of Random-k) for analysis; hence the derived results cannot well describe the real convergence performance of TopK-SGD. To this end, we first study the gradient distributions of TopK-SGD during training process through extensive experiments. We then theoretically derive a tighter bound for the Top-k operator. Finally, we exploit the property of gradient distribution to propose an approximate top-k selection algorithm, which is computing-efficient for GPUs, to improve the scaling efficiency of TopK-SGD by significantly reducing the computing overhead.

1 INTRODUCTION

Training large-scale deep neural networks (DNNs) generally exploits distributed synchronous stochastic gradient descent (SGD) optimization algorithms to reduce the overall training time. Let P be the number of workers in a distributed setting, and $\boldsymbol{x} \in \mathbb{R}^d$ denotes the model parameters with d dimensions. Then at the t-th iteration, distributed synchronous SGD updates the model parameters by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{g}_t^p, \tag{1}$$

where $\boldsymbol{g}_t^p \in \mathbb{R}^d$ is the stochastic gradient with its locally selected data for the loss function $f^p(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ and η_t is the learning rate. The aggregation of d-dimension gradients from P workers requires a communication complexity of $O(d)$[1], which generally limits the system scalability. Gradient sparsification (Strom, 2015; Dryden et al., 2016; Aji & Heafield, 2017; Chen et al., 2018; Lin et al., 2018) is a promising technique for distributed SGD, which can significantly reduce the communication traffic while reserving the model convergence. In gradient sparsification, a compressor $Comp_k$ is applied on each worker to locally select k, $k \leq d$, gradients for aggregation and $Comp_k \in \{\text{Top}_k, \text{Rand}_k\}$ (Stich et al., 2018). $Comp_k(\boldsymbol{g}_t^p) \in \mathbb{R}^d$ zeros out $(d-k)$ elements of $\boldsymbol{g}_t^p$ and keeps k elements unchanged. The zeroed-out $d - k$ elements are stored as residual $\boldsymbol{\epsilon}_t^p$ for the next iteration. Formally, the model parameters are updated by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \frac{1}{P} \sum_{p=1}^{P} Comp_k(\boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p) \text{ and } \boldsymbol{\epsilon}_{t+1}^p = \boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p - Comp_k(\boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p), \tag{2}$$

where $\boldsymbol{\epsilon}_t^p \in \mathbb{R}^d$ and $\boldsymbol{\epsilon}_0^p = \boldsymbol{0}$. In theory, distributed SGD with gradient sparsification (e.g., Top_k, Rand_k and any other k-contraction operators) with error compensation has been proved to have the

---

[1]The ring-based AllReduce collective can achieve the bandwidth optimal performance that is not related to the number of workers.

same order of convergence rate as vanilla SGD for both convex and non-convex problems if the number of iterations is large (Wangni et al., 2018; Stich et al., 2018; Alistarh et al., 2018; Jiang & Agrawal, 2018; Karimireddy et al., 2019; Tang et al., 2019; Zheng et al., 2019). The convergence rates are derived with a key contraction property of the sparsification operator $Comp_k$ (Top_k or $Rand_k$) (Stich et al., 2018; Alistarh et al., 2018), that is

$$\mathbb{E}_C[\|\boldsymbol{x} - Comp_k(\boldsymbol{x})\|^2] \leq (1 - k/d)\|\boldsymbol{x}\|^2, \forall \boldsymbol{x} \in \mathbb{R}^d, \tag{3}$$

where $\mathbb{E}_C$ is the expectation taking on the compressor and $\|\cdot\|$ is the ℓ_2-norm. For any $\boldsymbol{x} \in \mathbb{R}^d$, $Top_k(\boldsymbol{x}) \in \mathbb{R}^d$ selects the top k largest elements (in terms of the absolute value) of $\boldsymbol{x}$ with corresponding indices and sets other $d - k$ elements to zeros; while $Rand_k(\boldsymbol{x}) \in \mathbb{R}^d$ randomly (in a uniform distribution) selects k elements from $\boldsymbol{x}$ with corresponding indices and other $d-k$ elements are zeros. It is obvious that

$$\|\boldsymbol{x} - Top_k(\boldsymbol{x})\|^2 \leq \|\boldsymbol{x} - Rand_k(\boldsymbol{x})\|^2 \text{ and } \mathbb{E}_R[\|\boldsymbol{x} - Rand_k(\boldsymbol{x})\|^2] = (1 - k/d)\|\boldsymbol{x}\|^2. \tag{4}$$

Existing studies use the same error estimate for both Top_k and $Rand_k$ in distributed SGD by exploiting the properties of (4), which cannot differentiate the convergence behavior of two operators. In practice, however, TopK-SGD has a much faster convergence speed (in term of iterations) than SGD with $Rand_k$ (RandK-SGD) as empirically shown in (Stich et al., 2018). We also compare the convergence performance between TopK-SGD and RandK-SGD on a 16-worker distributed setting with three popular convolutional neural networks (VGG-16 (Simonyan & Zisserman, 2014), ResNet-20 and ResNet-50 (He et al., 2016)). Our results are shown in Fig. 1. We observe that TopK-SGD achieves very similar performance to the original distributed SGD (Dense-SGD), while RandK-SGD has much slower convergence than TopK-SGD. RandK-SGD even cannot converge on ImageNet. Therefore, though existing studies show that TopK-SGD and RandK-SGD have the same convergence bound, their theoretical results cannot explain the performance gap between TopK-SGD and RandK-SGD. Even some work (Karimireddy et al., 2019; Tang et al., 2019) exploits $\delta \leq 1$ to replace k/d in (3), they also fail to identify exact δ to distinguish Top_k and $Rand_k$.

(a) VGG-16 on CIFAR10   (b) ResNet-20 on CIFAR10   (c) ResNet-50 on ImageNet
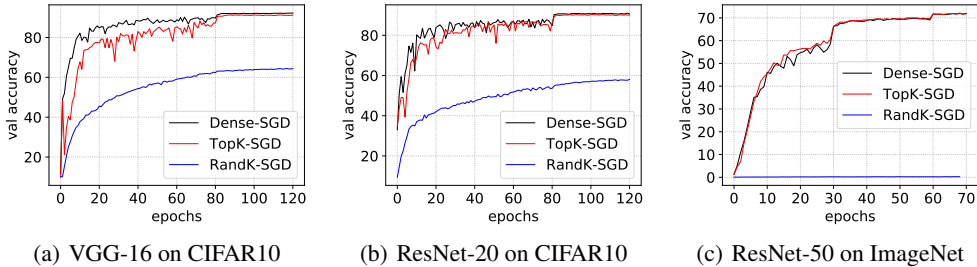
Figure 1: Convergence comparison between original distributed SGD (Dense-SGD), Top_k sparsification (TopK-SGD) and $Rand_k$ sparsification (RandK-SGD) at 16 distributed workers on the CIFAR10 (Krizhevsky et al., 2010) and ImageNet (Deng et al., 2009) data sets.

In this paper, we dive into the details of the Top_k operator in distributed SGD when training DNNs and provide a tighter bound than inequality (3) to explain the good convergence performance of TopK-SGD. The observation of gradients with Top_k sparsification further enables us to propose a new computational-efficient selection algorithm for gradient which preserves the convergence property. Our contributions are summarized as follows.

Contributions. (1) We empirically study the details of local stochastic gradients and observe that the elements of gradient follow Gaussian-like distributions through extensive experiments. (2) The Gaussian-like distribution enables us to intuitively explain that Top_k should have a much tighter bound than $Rand_k$, and we exploit the distribution property to formulate how Top_k outperforms $Rand_k$. (3) We design and implement an approximate top-k selection algorithm[2], which is much more efficient than existing top-k selection algorithms on GPUs. As compared with the existing sampling-based approximate top-k selection algorithm, we improve the scaling efficiency by 12-50% on our 16-GPU cluster.

---

[2] Our system implementation will be made open-source after the review process.

2 RELATED WORK

Gradient Quantization. In distributed training of neural networks, the communicated gradients can be quantized to low-bit precision (e.g., 16-bit (Micikevicius et al., 2018; Jia et al., 2018), 3-bit (Wen et al., 2017), 2.8-bit (Alistarh et al., 2017; Karimireddy et al., 2019) and even 1-bit (Seide et al., 2014; Strom, 2015)) while preserving nearly consistent convergence performance with the full precision (32-bit) counterpart. Recently general frameworks of gradient quantization with error compensation are proposed to generalize the theoretical results of low-bit communication (Wu et al., 2018; Jiang & Agrawal, 2018; Karimireddy et al., 2019; Tang et al., 2019; Haddadpour et al., 2019). However, the quantization method can only reduce the communication traffic in $32\times$ (i.e., 1-bit vs. 32-bit), and it could not be enough for large-scale models or low-bandwidth network connections.

Gradient Sparsification. Compared to gradient quantization, gradient sparsification is a much more promising communication traffic reduction technique as it can sparsify up to three orders of magnitude gradients be zero with little impact on the model convergence (Strom, 2015; Dryden et al., 2016; Aji & Heafield, 2017; Chen et al., 2018; Lin et al., 2018; Shi et al., 2019a). Due to the much success of gradient sparsification (e.g., Top-k sparsification) in significantly reducing the communication traffic in practice, much recent work tries to build theoretical guarantees for the SGD optimization algorithm with gradient specification (Wangni et al., 2018; Stich et al., 2018; Alistarh et al., 2018; Jiang & Agrawal, 2018; Shi et al., 2019b; Karimireddy et al., 2019; Tang et al., 2019). These theoretical frameworks try to generalize the sparsification operator with the bound of inequality (3) to derive the convergence results for SGD with gradient sparsification. However, the existing analysis fails to go insight into the details of gradient sparsification of Top_k which is much success applied in real-world application than other compression operators (e.g., Rand_k).

Gradient Distribution. To understand the behaviors of Top_k on stochastic gradients, we should study the gradient properties. Glorot & Bengio (2010) study the distribution of activation values of DNNs and also their corresponding gradients. They empirically showed that back-propagated gradients have Gaussian-like distribution, which helps understand the difficulty of training deep neural networks. A similar plot is shown in (Micikevicius et al., 2018), where the distribution of gradients helps analyze if the 16-bit representation of gradients would be overflow or underflow. These work has demonstrated that the gradients during training are likely located near zeros. We extend the similar studies on the gradient distribution for TopK-SGD.

3 STUDY ON STOCHASTIC GRADIENTS

### 3.1 GRADIENT DISTRIBUTION

In previous gradient sparsification studies (Strom, 2015; Dryden et al., 2016; Aji & Heafield, 2017; Chen et al., 2018; Lin et al., 2018), the basic rule of sparsification is to select "significant" elements of the gradients because they contribute more to the updates. The Top_k operator selects the exact local top-k elements of gradients so that it achieves nearly consistent convergence performance with Dense-SGD. Therefore, we would like to understand what is the difference between "significant" elements of the gradients and randomly selected elements of gradient. We conduct extensive experiments to study the gradient distributions on three areas of deep learning applications, including image classification, language modeling, and speech recognition. The selected models are: 1) **Feed-forward Neural Networks (FNNs)**. We build an FNN with three hidden fully connected layers (FNN-3) on the MNIST (LeCun, 1998) data set. 2) **Convolutional Neural Networks (CNNs).** We conduct experiments with LeNet-5 (LeCun et al., 2015) on MNIST, ResNet-20 (He et al., 2016) and VGG-16 (Simonyan & Zisserman, 2014) architectures on CIFAR10 (Krizhevsky et al., 2010). And 3) **Recurrent Neural Networks (RNNs).** We use Long Short Term Memory networks (LSTMs) on the Penn Treebank (PTB) (Marcus et al., 1993) and the AN4 (Acero, 1990) data sets. For PTB, we adopt a 2-layer LSTM model (LSTM-PTB) with 1500 hidden units per layer, and for AN4, we use a 5-layer LSTM model (LSTM-AN4) with 800 hidden units per layer.

The details of the experimental settings are shown in Table 1. As the compression operator is applied on the gradients, we first measure the distributions of the gradient's elements (histograms) on Dense-SGD. The results demonstrate the similar shapes as (Glorot & Bengio, 2010), while ours covers various applications (refer to Appendix A.1). Our interest is on TopK-SGD to check if gradients distributions perverse the same properties as Dense-SGD. During the training process of TopK-SGD

Table 1: Experimental settings. All models are trained by SGD with a 0.9 momentum. "BS" is the mini-batch size at each worker. "LR" is the initial learning rate which is decayed during training.

Type	Model	# Params	Weight Init.	Activation	BS	LR	Data Set
FNN	FNN-3	199,210	Xavier	ReLU	128	0.01	MNIST
CNN	LeNet-5	61,706	Xavier	ReLU	128	0.01	
	ResNet-20	269,722	Xavier, Kaiming	ReLU	32	0.1	CIFAR10
	VGG-16	14,728,266	Kaiming	ReLU	128	0.1	
RNN	LSTM-PTB	66,034,000	Uniform	Tanh	20	22	PTB
	LSTM-AN4	27,569,568	Xavier	Tanh	4	0.0002	AN4

($k = 0.001d$ for a d-dimension model), we measure the histograms of local gradients accumulated with the residuals (i.e., $\boldsymbol{u}_t^p = \boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p$). The histograms of $\boldsymbol{u}_t^1$ with different t on different models are shown in Fig. 2, where we only show the gradients from the first worker as different workers have very close gradient distributions. It is seen that different models have different shapes on the

(a) FFN-3   (b) LeNet-5   (c) ResNet-20

(d) VGG-16   (e) LSTM-PTB   (f) LSTM-AN4
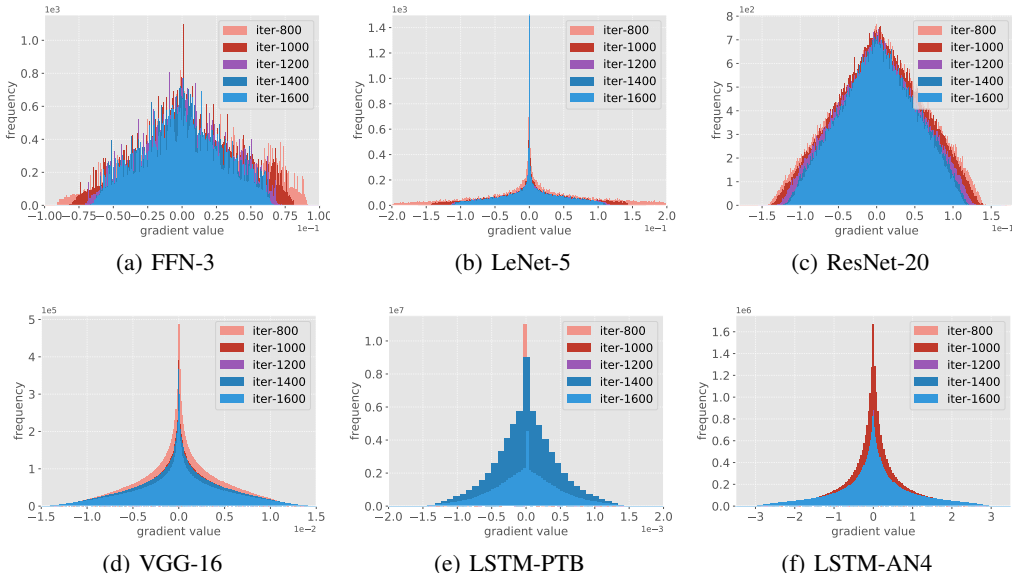
Figure 2: The histograms of $\boldsymbol{u}_t^1$ of TopK-SGD. For each model, the gradient histograms are plotted every 200 iterations from iteration 200 to 1600 (other iterations have similar shapes).

accumulated gradients, but one common feature is that most coordinates of $\boldsymbol{u}_t$ are close to zero. Compared to the full gradient SGD (Appendix A.1), TopK-SGD shows wider distributions, which could be mainly caused by residual accumulation. When selecting top-k largest values (in terms of absolute values) from $\boldsymbol{u}_t$, the selected values should be located at the left and right sides on the histograms. Therefore, performing Top_k on $\boldsymbol{u}_t$ should generate a vector whose ℓ_2-norm is very close to that of $\boldsymbol{u}_t$, that is $\|\text{Top}_k(\boldsymbol{u}_t)\|^2 \lesssim \|\boldsymbol{u}_t\|^2$. The intuitive result inspires us to formulate how much close of $\|\text{Top}_k(\boldsymbol{u}_t)\|^2$ to $\|\boldsymbol{u}_t\|^2$. Specifically, we would like to derive a variable $\gamma \leq (1 - k/d)$ such that $\|\boldsymbol{u}_t - \text{Top}_k(\boldsymbol{u}_t)\|^2 \leq \gamma \|\boldsymbol{u}_t\|^2$ holds.

3.2 THEORETICAL ANALYSIS AND RESULTS

We investigate the Top_k operator on $\boldsymbol{u}_t^p = \boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p$ (for ease of presentation, we use $\boldsymbol{u}$ to denote $\boldsymbol{u}_t^p$).

Error estimation of Top_k Operator. Let $\boldsymbol{\pi}$ denote a sorted vector of $|\boldsymbol{u}|/\|\boldsymbol{u}\|_\infty$ in descending order. That is $\boldsymbol{\pi}_{(i)} \geq \boldsymbol{\pi}_{(i+1)} \geq 0$ for $i = 1, 2, ..., d-1$, where $\boldsymbol{\pi}_{(i)}$ is the ith element of $\boldsymbol{\pi} \in \mathbb{R}^d$. Then we have

$$\frac{\|\boldsymbol{u} - \text{Top}_k(\boldsymbol{u})\|^2}{\|\boldsymbol{u}\|^2} = \frac{\|\boldsymbol{u} - \text{Top}_k(\boldsymbol{u})\|^2/\|\boldsymbol{u}\|_\infty}{\|\boldsymbol{u}\|^2/\|\boldsymbol{u}\|_\infty} = \frac{\|\tilde{\boldsymbol{u}} - \text{Top}_k(\tilde{\boldsymbol{u}})\|^2}{\|\tilde{\boldsymbol{u}}\|^2}, \tag{5}$$

where $\tilde{\boldsymbol{u}} = \boldsymbol{u}/\|\boldsymbol{u}\|_\infty$ and

$$\frac{\|\tilde{\boldsymbol{u}} - \text{Top}_k(\tilde{\boldsymbol{u}})\|^2}{\|\tilde{\boldsymbol{u}}\|^2} = \frac{\sum_{i=k+1}^d \boldsymbol{\pi}_{(i)}^2}{\sum_{i=1}^d \boldsymbol{\pi}_{(i)}^2}. \tag{6}$$

Assume that $\boldsymbol{u}_{(i)}$ follows a zero-mean normal distribution (e.g., Fig. 3(a)), then $\boldsymbol{\pi}$ is a decreasing function w.r.t. i as shown in Fig. 3(b). In order to evaluate Eq. (6), it is essential to calculate the area under the curve of $\boldsymbol{\pi}$. One can easily prove that $\boldsymbol{\pi}$ is convex and it is always less than the reference line ($y = -i/d + 1$) if $\boldsymbol{u}$ follows bell shaped distributions as illustrated in Fig. 2. Considering the areas of $A_1, A_2, A_3,$ and A_4 shown in Fig. 3(c), we have

$$\frac{\sum_{i=k+1}^d \boldsymbol{\pi}_{(i)}^2}{\sum_{i=1}^d \boldsymbol{\pi}_{(i)}^2} = \frac{A_1}{A_1 + A_2 + A_3}. \tag{7}$$

By simple calculation, we have

$$\frac{A_1}{A_1 + A_2 + A_3} \leq \frac{A_1 + A_4}{A_1 + A_2 + A_4} = \frac{\text{Area of } MDB}{\text{Area of } OCB} = \frac{\text{Area of } EBD}{\text{Area of } OAB} = \left(1 - \frac{k}{d}\right)^2, \tag{8}$$

where the second equality can be obtained from the similarity of triangle $\triangle MDB \sim \triangle COB$ and $\triangle EDB \sim \triangle AOB$, i.e.,

$$\frac{\text{Area of } MDB}{\text{Area of } OCB} = \frac{MD}{CO} = \frac{DB}{OB} = \frac{ED}{AO} = \frac{\text{Area of } EBD}{\text{Area of } OAB}. \tag{9}$$

Putting altogether, we have

$$\|\boldsymbol{u} - \text{Top}_k(\boldsymbol{u})\|^2 / \|\boldsymbol{u}\|^2 \leq (1 - k/d)^2 =: \gamma \tag{10}$$

and eventually

$$\|\boldsymbol{u} - \text{Top}_k(\boldsymbol{u})\|^2 \leq \gamma\|\boldsymbol{u}\|^2 \leq (1 - k/d)\|\boldsymbol{u}\|^2, \tag{11}$$

where $\gamma = (1 - k/d)^2$. The last inequality is always true as $|1 - k/d| \leq 1$. Our results can be
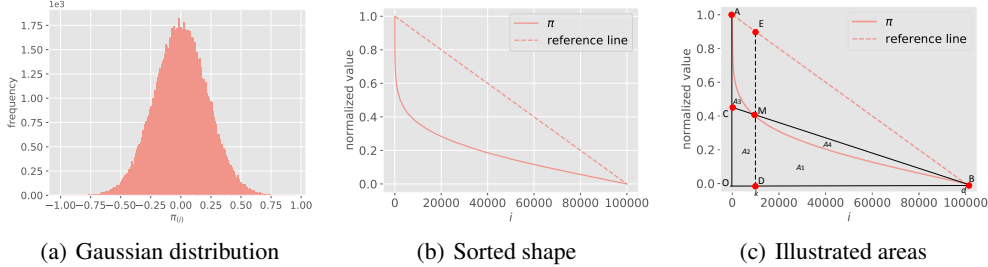

(a) Gaussian distribution   (b) Sorted shape   (c) Illustrated areas

Figure 3: The shape of $\boldsymbol{\pi}_{(i)}$ with different i with $d = 100,000$ and $\sigma = 1$.

summarized as the following theorem.

Theorem 1. *Assume that $\boldsymbol{u} \in \mathbb{R}^d$ follows a bell shaped distribution, then we have*

$$\|\boldsymbol{u} - Top_k(\boldsymbol{u})\|^2 \leq (1 - k/d)^2 \|\boldsymbol{u}\|^2. \tag{12}$$

Furthermore, it can be rearranged into the form that

$$\|\boldsymbol{u} - Top_k(\boldsymbol{u})\|^2 \leq (1 - \delta)\|\boldsymbol{u}\|^2, \quad \text{where } \delta = (2kd - k^2)/d^2. \tag{13}$$

Convergence Bound of TopK-SGD. We use the same assumptions on the objective function $f : \mathbb{R}^d \to \mathbb{R}$ as (Karimireddy et al., 2019). The assumptions are: 1) f is L-smooth and 2) f has a moment bound (i.e., $\mathbb{E}[\boldsymbol{g}] = \nabla f(\boldsymbol{x})$ and $\mathbb{E}[\|\boldsymbol{g}\|^2] \leq G^2$ for some $G > 0$, where $\boldsymbol{g}$ is a stochastic gradient and $\boldsymbol{x}$ is the model parameter). Therefore, we can directly use the the bound formulation of convergence rate with δ from (Karimireddy et al., 2019) in Remark 4.

Theorem 2. *If we set $\eta_t = \frac{1}{\sqrt{T+1}}$ for running TopK-SGD and under the assumptions of f, we have*

$$\min_{t \in [T]} \mathbb{E}[\|\nabla f(\boldsymbol{x}_t)\|^2] \leq \frac{4(f(\boldsymbol{x}_0) - f^*) + LG^2}{2\sqrt{T+1}} + \frac{4L^2 G^2 (1 - \delta)}{\delta^2 (T+1)}, \tag{14}$$

where f^* is the optimal solution.

The theorem indicates that after $T \geq O(1/\delta^2)$ iterations, the first term of the right-hand side of inequality (14) will dominate the bound so that the convergence rate becomes $O(1/\sqrt{T})$ which matches the rate of vanilla SGD. Note that our derived bound of $\delta = (2kd - k^2)/d^2$ is much tighter than k/d in previous studies (Stich et al., 2018; Alistarh et al., 2018; Jiang & Agrawal, 2018; Shi et al., 2019b; Karimireddy et al., 2019). Let $c = d/k$ denote the compression ratio of gradients. Previous results ($\delta = 1/c$) indicate that RandK-SGD or TopK-SGD should run after $T \geq O(c^2)$ iterations to make it catch up the convergence rate of Dense-SGD. Using inequality (11) for TopK-SGD, it just requires $T \geq O(c^4/(2c-1)^2)$ iterations to have the full gradient convergence rate. The result gives the explanation to why TopK-SGD can easily achieve nearly consistent convergence performance to Dense-SGD, while RandK-SGD could not (as shown in Fig. 1).

3.3 Gaussian$_k$: An Approximate Top$_k$ Operator

Though the Top$_k$ operator has good convergence property with significantly reduced communication size in distributed SGD, the top-k selection problem is not friendly to the current many-core processors like GPUs (Shanbhag et al., 2018). Inefficient Top$_k$ could make the overall wall-clock worse. For example, training a ResNet-50 (He et al., 2016) model on the ImageNet (Deng et al., 2009) data set on an Nvidia Tesla V100 GPU with a mini-batch size of 128 requires around 0.46 second per iteration[3]. When we distribute the training to 16 Tesla V100 GPUs connected with 10 Gbps Ethernet, the communication time of full gradients ($d = 25,557,032$) is around 0.2 second. However, the Top$_k$ operator with $k = 0.001d$ on the ResNet-50 model with the Tesla V100 GPU consumes 0.4 seconds. The 0.2-second communication overhead is saved, but it introduces another 0.4 second overhead at each iteration, which makes the training efficiency even worse. In DGC-SGD (Lin et al., 2018), the authors have noticed this problem, and they proposed to sample only 0.1% to 1% of the gradients to estimate the threshold hierarchically, which requires to invoke top-k selection twice on the subsets of the original vector. For ease of reference, we use DGC$_k$ to denote the hierarchical sampling method in selecting the largest top-k gradients.

Algorithm 1 Gaussian$_k$

Input: Stochastic gradients with residuals $\boldsymbol{u}_t^p$
Input: k and dimension d
1: Initialize $\hat{\boldsymbol{u}}$ as a zero vector with d dimensions;
2: μ, σ = mean and std of vector $\boldsymbol{u}_t^p$;
3: $p = 1 - k/d$;
4: $thres = \text{ppf}(\boldsymbol{u}_t^p, p, \mu, \sigma)$;
5: **for** $i = 0 \rightarrow 3$ **do**
6:     $masks = |\boldsymbol{u}_t^p| > thres$;
7:     $estimated_k$ = # of True values in $masks$;
8:     **if** $estimated_k < 2k/3$ **then**
9:        $thres = 0.5 \times thres$;
10:    **else if** $estimated_k > 4k/3$ **then**
11:       $thres = 1.5 \times thres$;
12:    **else**
13:       break;
14: $\hat{\boldsymbol{u}}[masks] = \boldsymbol{u}_t^p[masks]$;
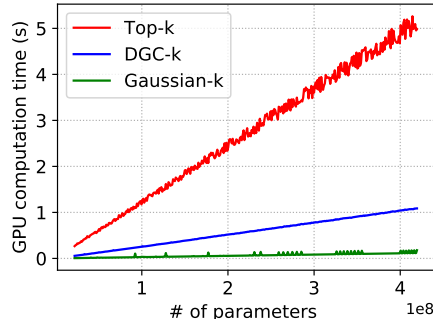15: Return $\tilde{\boldsymbol{u}}$;

Figure 4: The GPU computation time (lower is better) of Top$_k$, DGC$_k$ and Gaussian$_k$. We use the PyTorch tensor API, "tensor.topk()", for the Top$_k$ operator.

We propose an approximate Top$_k$ operator named Gaussian$_k$ by exploiting the Gaussian-like distribution property of gradients. The key ideas of Gaussian$_k$ are: 1) We regard the d-dimensional gradients (i.e., $\boldsymbol{u}_t^p$) at each iteration as a normal distribution with the mean (μ) and standard variance (σ) which can be directly calculated in an $O(d)$ complexity and the calculations are friendly to

---

[3]Note that the model is trained with the single-precision floating point (32-bit), so the Tensor Core of Tesla V100 GPU is not used.

GPUs. 2) We estimate the threshold by exploiting the percent point function (ppf) of $\boldsymbol{u}_t^p$ with three parameters: $p = 1 - k/d$, μ and σ. 3) As the distribution is not exactly normal, the ppf estimation could result in a threshold that could be slightly smaller or larger than the true threshold. We move to the estimated threshold to the left or right side several times such that we can have very close top-k largest absolute values. The algorithm of Gaussian$_k$ is shown in Algorithm 1.

4 EXPERIMENTS

As we mainly focus on gradient sparsification, we use the fp32 operations instead of exploiting lower precision for training models. The related software libraries are CUDA-10.1, cuDNN-7.5.0, NCCL-2.3.7, PyTorch-1.1.0, OpenMPI-4.0.1, and Horovod-0.16.4 (Sergeev & Balso, 2018).

4.1 NUMERICAL RESULTS OF THE TOP$_k$ OPERATOR

To validate the bound of inequality (11), we randomly (in Gaussian distribution) generate a $100,000$ dimension vector and compare the exact value of $\|\boldsymbol{u} - \text{Top}_k(\boldsymbol{u})\|^2/\|\boldsymbol{u}\|^2$ and $1 - k/d$ with ours derived $(1 - k/d)^2$. We also compare the three bounds in the real-world model training process. The results are shown in Fig. 5. It is seen that both ours and the previous result are in the upper side of the exact value, which indicates the derived bounds hold. With increased k, ours becomes better and better than the previous result. However, one may notice that the exact value is still much lower than ours. The reason is that our bound is derived by the reference line (Fig. 3(b)) but not the original function. Therefore, if the shape of $\boldsymbol{\pi}_{(i)}$ can be exactly formulated, one can derive a tighter bound for the Top$_k$ operator than $(1 - k/d)^2$ and we will leave this as our future work.
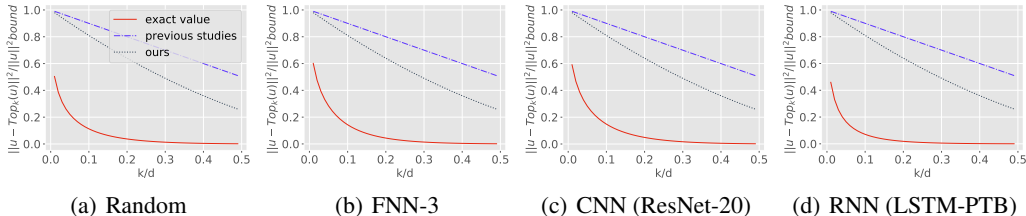

Figure 5: The comparison of bounds with a range of k.

4.2 GPU COMPUTATION EFFICIENCY OF SPARSIFICATION

To evaluate the computing efficiency of different top-k selection algorithms on GPUs, we conduct experiments on an Nvidia Tesla V100 GPU with d ranging from 20 million to 400 million and $k = 0.001d$. The GPU computation speed comparison between Top$_k$, DGC$_k$ and Gaussian$_k$ operators is shown in Fig. 4. For DGC$_k$, we use 1% as suggested in (Lin et al., 2018) to estimate the threshold. Note that tensor operations (e.g., top-k selection, mean and std calculations etc.) are from PyTorch's tensor APIs[4]. The experimental results show that the Top$_k$ operator becomes very slow with a large number of parameters, while Gaussian$_k$ only generates slight overheads. DGC$_k$ also becomes inefficient if d is large. It is crucial for the end-to-end training to have a computing-efficient operator on GPUs such that the extra computation overhead would not limit the system scalability.

4.3 CONVERGENCE PERFORMANCE OF GAUSSIANK-SGD.

To demonstrate the convergence performance of GaussianK-SGD, we run 120 epochs on CIFAR10 and 70 epochs on ImageNet with 16 workers. The top-1 validation accuracy of the evaluated models is shown in Fig. 6. Note that for each model, we use the same hyper-parameters for the three SGD algorithms. We can see that our GaussianK-SGD has nearly consistent validation accuracy with TopK-SGD, which indicates that our proposed Gaussian$_k$ operator can select close elements with Top$_k$. The gradient distributions in GaussianK-SGD are similar to TopK-SGD (Appendix A.2). In the evaluated three models, GaussianK-SGD and TopK-SGD have slight accuracy loss (around

---

[4]https://pytorch.org/docs/stable/tensors.html

0.6%-0.8%) compared to Dense-SGD. As suggested in (Lin et al., 2018), the small residuals could have staleness compared to the current gradients so that it could cause the slight accuracy loss. Some optimization tricks in (Lin et al., 2018) like momentum correction would address this problem.

(a) VGG-16 on CIFAR10   (b) ResNet-20 on CIFAR10   (c) ResNet-50 on ImageNet
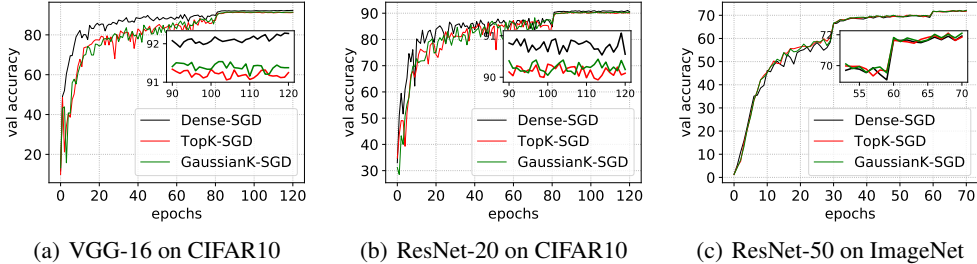
Figure 6: The convergence performance (top-1 validation accuracy) of distributed SGD with GaussianK-SGD using $k = 0.001d$ compared to TopK-SGD and Dense-SGD on 16 workers.

4.4 END-TO-END TRAINING SCALING EFFICIENCY OF GAUSSIANK-SGD.

We evaluate the average iteration time of GaussianK-SGD on the ImageNet (Deng et al., 2009) data set with four popular models (AlexNet (Krizhevsky et al., 2012), VGG-16 (Simonyan & Zisserman, 2014), ResNet-50 (He et al., 2016) and Inception-V4 (Szegedy et al., 2017)) on a 16-GPU cluster compared to Dense-SGD with full gradients, TopK-SGD with the original top-k selection and DGC-SGD (Lin et al., 2018) with hierarchical sampling. The cluster has four nodes connected with 10 Gbps Ethernet (10GbE), and each node contains four Nvidia Tesla V100 GPUs (the PCIe version with 32GB memory). We also set $k = 0.001d$ for all the sparsified algorithms. The results are shown in Table 2, which shows that TopK-SGD is even slower than Dense-SGD on the 16-GPU cluster, while our GaussianK-SGD runs much faster than Dense-SGD, TopK-SGD, and DGC-SGD. Specifically, GaussianK-SGD is $1.19\times$-$2.33\times$ faster than Dense-SGD, $1.36\times$-$3.63\times$ faster than TopK-SGD, and $1.11\times$-$1.51\times$ faster than DGC-SGD, respectively. Even on the VGG-16 model, which has several large-size fully connected layers, GaussianK-SGD can achieve 85.5% scaling efficiency on the 16-GPU cluster with low-bandwidth Ethernet.

Table 2: Wall-clock time of end-to-end training with ImageNet on 16 Tesla V100 GPUs connected with 10 GbE. The batch size for each GPU is 128, and the input image resolution is 224×224.

Model	# Params (Million)	Iteration Time (s)				Scaling Efficiency (%)			
		Dense-S.	TopK-S.	DGC-S.	Gaussian-S.	Dense-S.	TopK-S.	DGC-S.	Gaussian-S.
AlexNet	61	0.571	0.891	0.369	**0.245**	14.1	9.0	21.8	**32.8**
VGG-16	138	2.068	3.010	1.540	**1.311**	54.2	37.2	72.8	**85.5**
ResNet-50	25	0.699	0.810	0.655	**0.586**	65.8	56.8	70.2	**78.5**
Inception-V4	42	1.022	1.268	0.916	**0.787**	67.5	54.4	75.3	**87.7**

5 CONCLUSION

In this paper, we first identified that existing theoretical results fail to explain the convergence performance of distributed SGD algorithms with Top-k gradient sparsification (TopK-SGD). Then we empirically studied gradient distributions during training with TopK-SGD through extensive experiments, and observe that the elements of stochastic gradients are mostly located near zero (Gaussian-like distribution). The observation enables us to build a theoretically tighter bound for the Top_k operator, which makes the convergence property of TopK-SGD explainable. According to the distribution of gradients, we propose an approximate top-k selection algorithm named Gaussian_k which is much efficient than the existing top-k selection algorithms on GPUs. We finally conduct extensive experiments to verify our derived bound for the Top_k operator and the convergence performance of distributed SGD with Gaussian_k (GaussianK-SGD). In terms of the scaling efficiency, GaussianK-SGD achieves up to $2.33\times$, $3.63\times$ and $1.51\times$ faster training speed than full gradient SGD, TopK-SGD and DGC-SGD on a 16-GPU cluster connected with 10 Gbps Ethernet, respectively.

REFERENCES

Alejandro Acero. Acoustical and environmental robustness in automatic speech recognition. In *Proc. of ICASSP*, 1990.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 440–445, 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. 1709–1720, 2017.

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*, pp. 5977–5987, 2018.

Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakr-ishnan. AdaComp: Adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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.

Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pp. 1–8. IEEE, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Trading redundancy for communication: Speeding up distributed SGD for non-convex optimization. In *International Conference on Machine Learning*, pp. 2545–2554, 2019.

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.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pp. 2530–2541, 2018.

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*, pp. 3252–3261, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www.cs.toronto.edu/kriz/cifar. html*, 2010.

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. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann.lecun.com/exdb/lenet*, 20:5, 2015.

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*, 2018.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *International Conference on Learning Representations*, 2018.

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 *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

Anil Shanbhag, Holger Pirk, and Samuel Madden. Efficient top-k query processing on massively parallel hardware. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1557–1570. ACM, 2018.

Shaohuai Shi, Qiang Wang, Kaiyong Zhao, Zhenheng Tang, Yuxin Wang, Xiang Huang, and Xiaowen Chu. A distributed synchronous SGD algorithm with global Top-k sparsification for low bandwidth networks. In *The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)*, pp. 2238–2247, 2019a.

Shaohuai Shi, Kaiyong Zhao, Qiang Wang, Zhenheng Tang, and Xiaowen Chu. A convergence analysis of distributed SGD with communication-efficient gradient sparsification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3411–3417, 2019b.

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

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pp. 4452–4463, 2018.

Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pp. 6155–6165, 2019.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1306–1316, 2018.

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. 1509–1519, 2017.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized SGD and its applications to large-scale distributed optimization. *International Conference on Machine Learning*, 2018.

Shuai Zheng, Ziyue Huang, and James T Kwok. Communication-efficient distributed blockwise momentum sgd with error-feedback. In *Advances in neural information processing systems*, 2019.

A APPENDIX

A.1 GRADIENT DISTRIBUTION ON DENSE-SGD
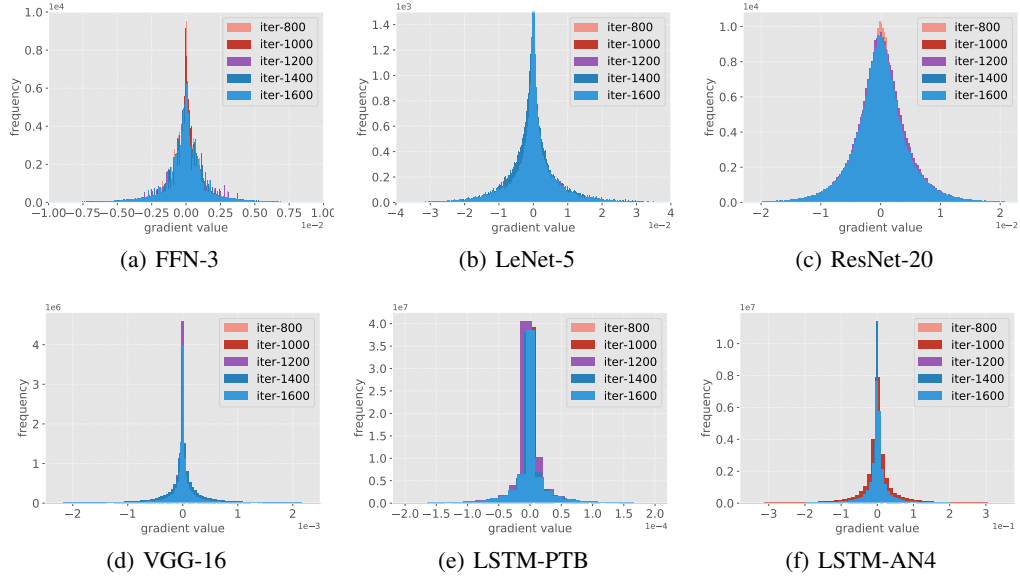
Figure 7: The histograms of $\boldsymbol{u}_t^1$ during Dense-SGD training process.

A.2 GRADIENT DISTRIBUTION ON GAUSSIAN-SGD
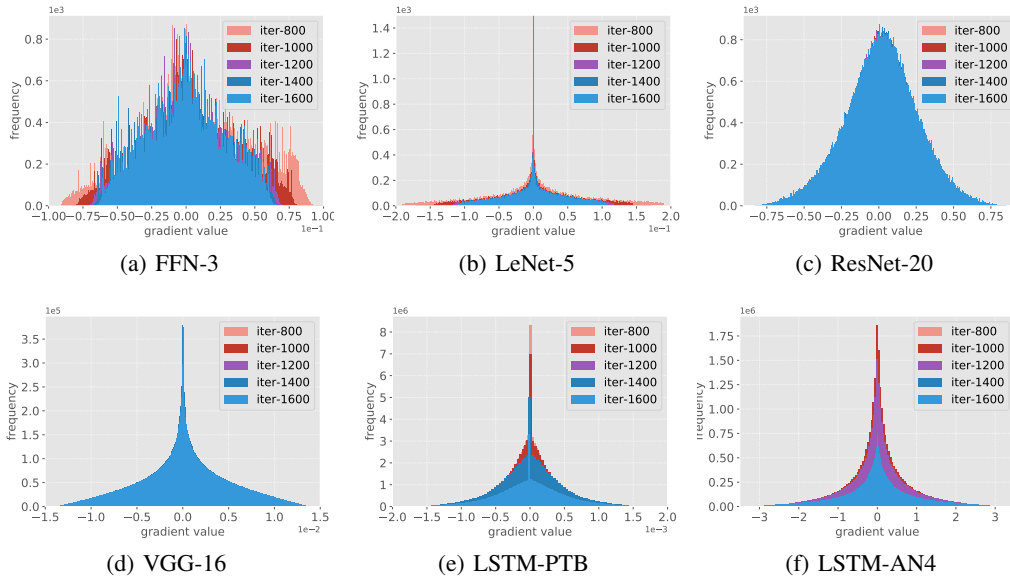
Figure 8: The histograms of $\boldsymbol{u}_t^1$ during GaussianK-SGD training process.