Don't Waste Your Bits! Squeeze Activations and Gradients for Deep Neural Networks via TINYSCRIPT

Fangcheng Fu [1 2] **Yuzheng Hu** [1] **Yihan He** [1] **Jiawei Jiang** [3] **Yingxia Shao** [4] **Ce Zhang** [3] **Bin Cui** [1 5]

Abstract

Recent years have witnessed intensive research interests on training deep neural networks (DNNs) more efficiently by quantization-based compression methods, which facilitate DNNs training in two ways: (1) activations are quantized to shrink the memory consumption, and (2) gradients are quantized to decrease the communication cost. However, existing methods mostly use a uniform mechanism that quantizes the values evenly. Such a scheme may cause a large quantization variance and slow down the convergence in practice.

In this work, we introduce TINYSCRIPT, which applies a non-uniform quantization algorithm to both activations and gradients. TINYSCRIPT models the original values by a family of Weibull distributions and searches for "quantization knobs" that minimize quantization variance. We also discuss the convergence of the non-uniform quantization algorithm on DNNs with varying depths, shedding light on the number of bits required for convergence. Experiments show that TINYSCRIPT always obtains lower quantization variance, and achieves comparable model qualities against full precision training using 1-2 bits less than the uniform-based counterpart.

1. Introduction

Deep learning (DL) has brought revolutionary changes to machine learning and made remarkable improvements on various complicated tasks such as image recognition, natural language processing, recommendation, and autonomous driving (He et al., 2016; Devlin et al., 2019; Arun & Govindan, 2018; Chen et al., 2019; Zhang et al., 2020; Gharibshah et al., 2020). Recently, with the incredible surge of big data and trend of edge-computing, many researchers have devoted efforts to training deep neural networks (DNNs) on edge devices. However, due to the large amount of data and increasingly complex model architectures, training DNNs becomes both **memory-** and **communication-bounded**. For one thing, activations (a.k.a. feature maps) in forward propagation need to be stored for gradient computation in backward propagation, which leads to large memory footprints. For another, gradients need to be sent to central servers or exchanged among workers for model update, resulting in heavy network transmission.

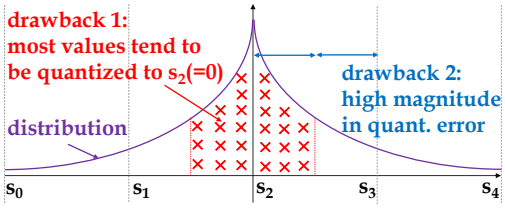


Figure 1. Uniform quantization with $n = 4$ on a distribution which has the properties of central tendency and long-tailed.

One approach to alleviating the burden of memory consumption and communication overhead is reducing the data size by quantizing values to fewer bits, at the price of precision loss. For instance, gradient quantization accelerates the overall training speed since the communication time is decreased (Alistarh et al., 2017; Wen et al., 2017), whilst (Chakrabarti & Moseley, 2019) copes with the memory bottleneck by quantizing activations into 4 bits.

Given a tensor $\boldsymbol{x}$ and a set of quantization points $\boldsymbol{s} = \{s_t\}_{t=0}^{n}$ satisfying $-M = s_0 < s_1 < ... < s_n = M$, where $M = ||\boldsymbol{x}||_\infty$ acts as a scaling factor[1], the stochastic quantization on each element $x_i \in [s_t, s_{t+1})$ is defined as

$$\tilde{x}_i = \begin{cases} s_t, & \text{with probability } (s_{t+1} - x_i)/(s_{t+1} - s_t) \\ s_{t+1}, & \text{with probability } (x_i - s_t)/(s_{t+1} - s_t) \end{cases}.$$

In a sense, each element is more likely to be quantized

[1]Department of Computer Science and Technology & Key Laboratory of High Confidence Software Technologies (MOE), Peking University [2]Tencent Inc. [3]ETH Zurich [4]Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, BUPT [5]Center for Data Science & National Engineering Laboratory for Big Data Analysis and Applications, Peking University. Correspondence to: Bin Cui <bin.cui@pku.edu.cn>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1](Alistarh et al., 2017) scales gradient with L_2 norm in their analysis, but uses L_∞ norm in practice.

to the nearest point, but the stochastic technique ensures unbiasedness, i.e., $\mathbb{E}[\tilde{\boldsymbol{x}}] = \mathbb{E}[\boldsymbol{x}]$. Although unbiased, the errors between the quantized and original values, which we refer to as *quantization variance*, can significantly affect the convergence. More aggressive quantization (smaller n) results in a higher variance, which inevitably slows down the convergence or even leads to divergence. To obtain better convergence, one may use more quantization levels (larger n) to reduce the variance or use a smaller step size to restrict the effect of variance. Nonetheless, such approaches cannot completely solve the problem and will either lower the compression rate or require more training steps.

We notice that existing works for DNNs essentially adopt a set of uniform quantization points, i.e., they evenly split the value range into n intervals. Nonetheless, empirical results have revealed that activation/gradient values conform to a unimodal and symmetric distribution rather than a uniform distribution (Bernstein et al., 2018). Worse, the value distribution usually has the properties of *central tendency* and *long-tailed* in practice, as we will describe later. Obviously, the stark difference between the uniform assumption and real distribution is the root cause of the high variance. As summarized in Figure 1, there are two drawbacks when adopting uniform quantization — (1) a large amount of coordinates with small absolute values are more likely to be quantized to 0, which causes information loss; (2) the range of each interval is large due to the long tail property, leading to high magnitude of quantization error.

Owing to the fundamental limitations in existing works, we try to study this problem from a different angle: *can we find the optimal quantization points with minimum quantization variance based on the value distribution?* Motivated as such, this work proposes a novel, distribution-aware quantization framework for DNNs called TINYSCRIPT, which leverages the distribution property to achieve optimal quantization. To summarize, we list the main contributions as below:

* TINYSCRIPT is superior to previous works in two ways: (1) previous works focus on either activations or gradients, whilst TINYSCRIPT is a unified framework that considers both; (2) TINYSCRIPT adopts a non-uniform quantization which fits values with Weibull priors and computes optimal quantization points.

* In order to interpret the rationality of adopting Weibull prior, we first discuss the distribution of activations and gradients using an MLP model. Then we analyze the convergence of SGD with quantization and build a bridge between the convergence and depth of neural network.

* Empirical results reveal that TINYSCRIPT achieves similar convergence performance as full precision training with approximately 2.3-3.2 bits, which gives 0.8-1.8 bits improvement over the uniform-based counterpart.

2. Related Works

In this section, we briefly go through the prior researches and preliminary literature related to our topic.

Weibull-related Random Variables Let random variable (r.v.) X with PDF $p(x) = \frac{k}{2\lambda}(\frac{|x|}{\lambda})^{k-1}\exp(-(\frac{|x|}{\lambda})^k)$ be a *double-Weibull* r.v. parameterized by shape $k > 0$ and scale $\lambda > 0$, which is extended from the Weibull distribution. A r.v. Y satisfying $\mathbb{P}(|Y| \geq y) \leq \alpha\exp(-(\frac{y}{\beta})^{\frac{1}{\theta}})$ for all $y > 0$ and for some $\theta > 0$ is called a *sub-Weibull* r.v. with tail θ, denoted by $Y \sim \text{subW}(\theta)$ (Vladimirova et al., 2018). Note that $X \sim \text{subW}(\frac{1}{k})$ since $\mathbb{P}(|X| \geq x) = \exp(-(\frac{x}{\lambda})^k)$.

Low-memory Training Various works have been developed to reduce the memory footprint of activations. A popular technique is to swap-out activations from GPU memory to CPU memory during forward pass and swap-in during backward pass (Rhu et al., 2016; Wang et al., 2018b). Our work is orthogonal to these works. One can first quantize the activations and then swap-out in order to reduce IO overhead of swapping. Furthermore, the swapping technique does not fit edge devices since there is no sufficient host memory. (Chakrabarti & Moseley, 2019) adopts the quantization technique in training DNNs, which is similar to ours, but only focuses on uniform quantization and conforms to 4 bits. In contrast, we leverage the distribution property and derive a more advanced non-uniform quantization method.

Gradient Compression Since distributed training is common nowadays, numerous researches try to quantize gradients into smaller sizes (Wen et al., 2017; Alistarh et al., 2017; Wu et al., 2018; Mishchenko et al., 2019). The most notable gradient quantization works for DNNs are TernGrad and QSGD. TernGrad proves the convergence of gradient quantization methods. However, it only considers an extreme case with $n = 2$. QSGD discusses the trade-off between n and convergence, but it merely focuses on uniform quantization. Note that TernGrad is a special case for both TINYSCRIPT and QSGD, since $\boldsymbol{s} = [-M, 0, M]$ if $n = 2$.

Another line of research focuses on how to sparsify the gradients by only sending large values and dropping the small ones (Stich et al., 2018; Wangni et al., 2018; Wang et al., 2018a; Sun et al., 2019). These approaches can highly reduce the size of gradients. Our work is orthogonal to the sparsification methods. Since values conform to a unimodal and symmetric distribution, there is a large portion of values quantized to the middle point (in most cases, zero) if we use fewer bits. Therefore, the quantized tensor is automatically sparsified. Moreover, since sparsification accumulates the filtered values for further updates, the memory consumed by activations even increases. So whether the sparsification technique is applicable for activations is still unknown. Consequently, we focus on the quantization-based methods in this work and leave the sparsification as future work.

Low-precision training A number of works have focused on training with low-precision values, e.g., 8-bit fixed-point numbers (Drumond et al., 2018; Banner et al., 2018; Wang et al., 2018c; Cambier et al., 2020). Empirical results have shown that such approaches are suitable for low-power specialized hardware that supports low-precision arithmetic. Nevertheless, our work differs from these methods — we only quantize activations and gradients and all computations are carried out in full precision. As we will describe later, quantization error does not propagate across layers in DNNs. Hence, our method can use fewer bits in quantization. To the best of our knowledge, there is no 4-bit floating-point representation since there will be insufficient bits for mantissa and exponent. As a result, we do not compare with this kind of works due to the difference in goals.

Other Non-uniform Quantization There are also works that adopt non-uniform quantization. The most distinguishable technique is using quantiles. ZipML (Zhang et al., 2017) formulates the problem of searching quantization points to reduce variance, which is similar to our work. However, a heuristic algorithm based on histograms is adopted to solve the problem, which actually results in quantiles. SketchML (Jiang et al., 2018; 2020) uses a data structure called quantile sketch to approximate the quantiles (Greenwald et al., 2001; Li & Li, 2018). In contrast, our approach derives the optimal quantization points under certain distribution rather than quantiles. Moreover, these works mainly focus on generalized linear models (GLMs) rather than DNNs. The reason is that the procedure of locating quantiles processes the values sequentially and involves complex operations to update the data structures. Thus it is infeasible for DNNs where the activations/gradients have a much larger number of elements than GLMs.

LQ-Nets (Zhang et al., 2018) focuses on training QNNs by learning quantizers. Specifically, it learns an encoding for each coordinate and computes the quantization points via the encoding. However, since storing the encoding requires extra memory, it takes 2K bits in K-bit quantization when training. Hence, it does not match our goal of squeezing memory. Furthermore, the update procedure of the encoding caused non-trivial overhead, as described in their paper. To the contrary, we propose to fit the value distribution by a reasonable prior and search the optimal quantization points by minimizing the variance. By this means, our method does not require extra memory and has the same time complexity as the vanilla uniform quantization.

3. Methodology

In this section, we first formulate the problem of minimizing quantization variance, and then introduce the proposed distribution-aware quantization framework, namely TINYSCRIPT.

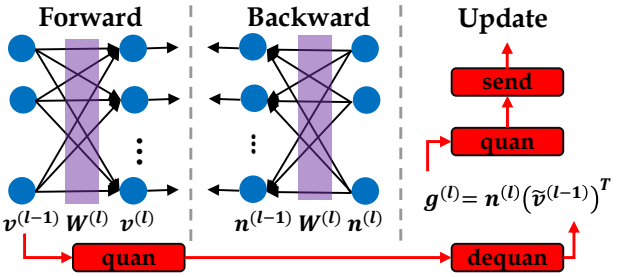


Figure 2. Illustration of quantization for activations and gradients on a fully connected layer with depth l.

3.1. Overview and Problem Formulation

Figure 2 illustrates the quantization for activations and gradients with an example of fully connected layer. In forward pass, the l-th layer takes as input the activations $\boldsymbol{v}^{(l-1)}$ from previous layer and generates $\boldsymbol{v}^{(l)}$ to the next layer. Then $\boldsymbol{v}^{(l-1)}$ is quantized to reduce memory usage. In backward pass, in addition to propagating derivatives $\boldsymbol{n}^{(\cdot)}$, gradients $\boldsymbol{g}^{(l)}$ are computed with the quantized activation $\tilde{\boldsymbol{v}}^{(l-1)}$. Upon model update, we quantize $\boldsymbol{g}^{(l)}$ to reduce network transmission, and the final update will be performed with the quantized gradient $\tilde{\boldsymbol{g}}^{(l)}$.

Obviously, the quantization strategy only perturbs the computation of gradients ($\boldsymbol{g}$), whilst forward and backward units ($\boldsymbol{v}$ and $\boldsymbol{n}$) are unaffected. It ensures that the error caused by quantization will not be accumulated as the network goes deeper. To limit the impact brought by quantization, our goal is to *minimize the expected quantization variance* $\mathbb{E}||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_2^2$ for a given tensor $\boldsymbol{x}$, motivated by the facts that quantization variance on $\boldsymbol{g}$ can be reckoned as part of stochastic variance and $||\boldsymbol{n}\tilde{\boldsymbol{v}}^T - \boldsymbol{n}\boldsymbol{v}^T||_F = ||\boldsymbol{n}||_2||\tilde{\boldsymbol{v}} - \boldsymbol{v}||_2$. Suppose the size of $\boldsymbol{x}$ is D and each value in $\boldsymbol{x}$ conform to some distribution $p(x)$, we have

$$\begin{aligned}\mathbb{E}_p||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_2^2 &= D\int_{s_0}^{s_n} p(x)\mathbb{E}(\tilde{x} - x)^2 dx \\ &= D\sum_{t=0}^{n-1}\int_{s_t}^{s_{t+1}} p(x)(s_{t+1} - x)(x - s_t)dx.\end{aligned} \tag{1}$$

Given the number of quantization levels n, an ideal quantization mechanism is expected to minimize Equation 1. There are two critic issues: (1) a good estimation for the distribution $p(x)$, and (2) a method to search quantization points $\boldsymbol{s}$ for a given $p(x)$. In the rest of the paper, we assume tensors to be quantized are flattened as vectors and $||\cdot||$ denotes the L_2 norm for simplicity.

3.2. Modeling of Distribution

The distribution $p(x)$ is vital to the minimum expected variance achieved by the optimal $\boldsymbol{s}$. An initial thought is to fit

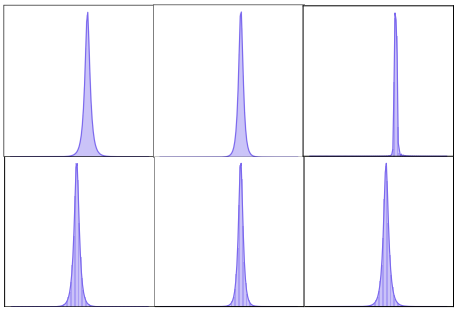


Figure 3. Histograms and fitted curves of activations (top row) and gradients (bottom row) of randomly chosen `conv` layers and training steps when training WideResNet34 on Cifar10. x-axis represents value range and y-axis represents counts of values.

$p(x)$ using methods like kernel density estimation or empirical distribution. However, it would be expensive if we fit $p(x)$ on-the-fly. Worse, the empirical distribution is usually non-differentiable, making the searching of $\boldsymbol{s}$ infeasible. As a result, we determine to assume a prior on $p(x)$. In each iteration, we compute statistics of the tensor, such as mean and standard deviation, and then estimate $p(x)$ with the prior. It requires scanning only one pass on the tensor and can be easily parallelized with SIMD instructions.

Nevertheless, it needs careful investigation to choose a good prior. One may guess values are Gaussian distributed due to the Central Limit Theorem, whereas it is not true in almost all cases. (Bernstein et al., 2018) concludes that gradient values are unimodal and symmetric, but they focus on the convergence of their algorithm rather than formulating the distribution. We plot some empirical distribution of activations and gradients in Figure 3, and notice that in addition to unimodal and symmetric, $p(x)$ is central tendency and long-tailed. Inspired by this, we propose to model values with the double-Weibull distribution [2]:

$$p(x) = \frac{k}{2\lambda}\left(\frac{|x|}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{|x|}{\lambda}\right)^{k}\right), 0 < k <= 1.$$

The restriction on k guarantees $p(x)$ is monotonic on negative or positive side. The parameter k also controls the shape of $p(x)$ — for a smaller k, $p(x)$ is more central tendency and has a longer tail. Therefore $p(x)$ is able to express distributions with various tails. In Section 4.1, we will discuss the rationality of the modeling in-depth.

In order to fit the distribution efficiently, we develop a moment-based method to approximate the distribution with mean and standard deviation (stddev). We present the method in Algorithm 1 and go through its details below.

[2]For activations after ReLU, we only consider the positive side and double the quantization level n. We also exclude the zeroes when we compute the mean and standard deviation.

Algorithm 1 Moment-based distribution fitting

Input: tensor $\boldsymbol{x}$
Output: fitted distribution

1: **function** INITCVTABLE(table step $delta = 0.001$)
2: Initialize empty table T
3: **for** $k \leftarrow 0.1$ to 1 by $delta$ **do**
4: $CV \leftarrow \sqrt{\Gamma(1+2/k)/\Gamma^2(1+1/k) - 1}$
5: Insert $\langle k, CV\rangle$ to T
6: **return** T
7: **function** MOMENTESTIMATE(stats $stat$, table T)
8: $\mu, \sigma \leftarrow |stat.mean|, stat.stddev$
9: Lookup $\langle k', CV'\rangle$ in T with binary search such that $|\sigma/\mu - CV'|$ is the smallest
10: $\lambda' \leftarrow \mu/\Gamma(1+1/k')$
11: **return** $\langle k', \lambda'\rangle$
12: **if** T not initialized **then**
13: $T \leftarrow$ INITCVTABLE()
14: Compute positive and negative statistics for tensor $\boldsymbol{x}$
15: $pos \leftarrow$ MOMENTESTIMATE($\boldsymbol{x}.pos\_stat, T$)
16: $neg \leftarrow$ MOMENTESTIMATE($\boldsymbol{x}.neg\_stat, T$)
17: **return** $\langle pos, neg\rangle$

Since moments $\mathbb{E}(X^t) = \lambda^t\Gamma(1+t/k)$, where $\Gamma(\cdot)$ is the gamma function, the coefficient of variation (CV) satisfies

$$CV = \frac{\sigma}{\mu} = \frac{\sqrt{\mathbb{E}(X^2) - (\mathbb{E}X)^2}}{\mathbb{E}X} = \sqrt{\frac{\Gamma(1+2/k)}{\Gamma^2(1+1/k)} - 1},$$

where μ, σ are mean and stddev. Therefore, we compute a $\langle k, CV\rangle$ lookup table beforehand (line 1-6). Given a tensor, we fit two distributions for positive and negative sides individually (line 14-16). Once CV is computed, we lookup the closest one from the table with a corresponding k' (line 9). Furthermore, since CV is monotonic decreasing w.r.t. k, the lookup can be executed with binary search efficiently. Finally, we can compute λ' inspired by the first-order moment $\mathbb{E}(X) = \lambda\Gamma(1+1/k)$ (line 10).

3.3. Searching of Optimal Quantization

Although we have modeled values via a Weibull prior, whether the minimum of Equation 1 exists in the domain of quantization points $\boldsymbol{s}$ is still unknown. If yes, how to find the optimal $\boldsymbol{s}^*$ that minimizes Equation 1 for any distribution $\langle k, \lambda\rangle$ and level n remains unsolved.

To answer these questions, we formulate the searching of $\boldsymbol{s}$ that achieves minimum variance as an optimization problem. For simplicity, we make the following adjustments: (1) we only consider the case of $\lambda = 1$, whereas we can easily migrate the discussion to arbitrary λ with a scaling factor; (2) we leverage the symmetry in distribution to assume n to be an even number (so that zero is always a quantization point)

and only consider the non-negative side; (3) we assume there is a sufficiently large value M ($M < \infty$) such that all (absolute) values are not greater than M. Now the objective we study becomes the expected quantization variance on the non-negative side:

$$V_{\hat{n}}(\hat{\boldsymbol{s}}) \coloneqq \sum_{t=0}^{\hat{n}} \int_{\hat{s}_t}^{\hat{s}_{t+1}} kx^{k-1} \exp\left(-x^k\right)(\hat{s}_{t+1}-x)(x-\hat{s}_t)dx,$$

where $\hat{n} = n/2$ and $\{\hat{\boldsymbol{s}}\}_{t=0}^{\hat{n}} = \{\boldsymbol{s}\}_{t=\hat{n}}^{n}$.

Thus, the searching of optimal quantization points can be turned into the following problem:

$$\arg\min_{\hat{\boldsymbol{s}}} V_{\hat{n}}(\hat{\boldsymbol{s}}) \text{ s.t. } 0 = \hat{s}_0 < \hat{s}_1 < ... < \hat{s}_{\hat{n}} = M.$$

Theorem 1. *There is a unique stationary point $\hat{\boldsymbol{s}}^*$ of $V_{\hat{n}}$ in the domain of $\hat{\boldsymbol{s}}$, and $V_{\hat{n}}$ attains minimum only at $\hat{\boldsymbol{s}}^*$.*

Theorem 1 implies that the minimum quantization variance is attained at a unique stationary point. As a result, we can use gradient descent to solve the optimization problem. The partial derivative of $\hat{s}_t (t = 1, 2, ..., \hat{n} - 1)$ is given below:

$$\begin{aligned}\frac{\partial V_{\hat{n}}}{\partial \hat{s}_t} =& -\Gamma(1+\frac{1}{k}, \hat{s}_{t+1}^k) + \Gamma(1+\frac{1}{k}, \hat{s}_{t-1}^k) \\ &+ \hat{s}_{t+1}\exp(-\hat{s}_{t+1}^k) - \hat{s}_{t-1}\exp(-\hat{s}_{t-1}^k) \\ &- (\hat{s}_{t+1} - \hat{s}_{t-1})\exp(-\hat{s}_t^k)\end{aligned} \tag{2}$$

where $\Gamma(\cdot,\cdot)$ denotes the incomplete gamma function. A general method of searching optimal quantization points can be: (1) randomly initialize $\hat{\boldsymbol{s}}$, e.g., uniform quantization points; (2) iteratively update $\hat{\boldsymbol{s}}$ via gradient descent by Equation 2 until converged.

Algorithm 2 shows how we compute the quantization points on-the-fly. Optimal points are pre-computed (line 1-8). Motivated by three-sigma rule of thumb, we set $M = 3\sigma$ heuristically (line 5). For tensor $\boldsymbol{x}$, we first fit distributions by Algorithm 1, then we lookup the optimal points and scale by λ (line 9-11). Finally the quantization points are reorganized in ascending order (line 16).

3.4. Implementation and Complexity Analysis

To implement TINYSCRIPT, we customize forward and backward propagation with PyTorch (Paszke et al., 2019), and we leverage NVIDIA Thrust library to perform statistics reduction and parallel quantization. For practical concerns, we further make two adjustments as described below.

First, we follow (Chakrabarti & Moseley, 2019) to integrate quantization of activations with the checkpointing technique (Chen et al., 2016). For instance, we regard consecutive `bn-relu-conv` chain as one block and only quantize and store one activation (input of `bn`), whilst the rest (inputs of `relu` and `conv`) are re-computed in backward pass. Such an approach is worthy since the time cost for `bn-relu` is much smaller than `conv` and is able to reduce both memory consumption and quantization overhead. In practice, we store the batch-wise mean and stddev in full precision since they are small in size. Therefore, it is obvious that the re-computed outputs of `bn` are identical to the directly quantized version in expectation. Consequently, in our following analysis, we simply assume all activations are quantized rather than re-computed.

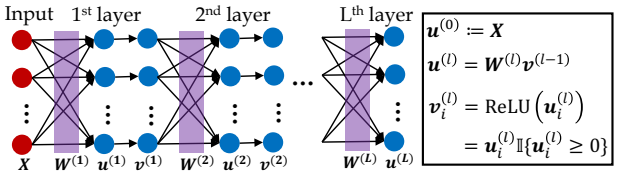
Algorithm 2 Quantization Calculation

Input: fitted distribution $\langle pos, neg\rangle$, min and max values $\boldsymbol{x}.min$, $\boldsymbol{x}.max$, #intervals n
Output: Optimal quantization points

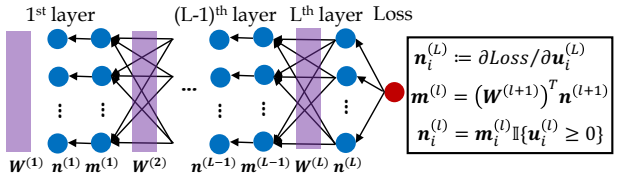
1: **function** INITQUANTABLE(max allowed $\hat{n}_{max} = 8$)
2: Initialize empty table Q
3: **for** $k \leftarrow T$.keys() **do**
4: **for** $\hat{n} \leftarrow 2, 3, ..., \hat{n}_{max}$ **do**
5: $M \leftarrow 3\sqrt{\Gamma(1+2/k) - \Gamma^2(1+1/k)}$
6: Iteratively compute $\{\hat{\boldsymbol{s}}^*\}_{t=1}^{\hat{n}-1}$ by Equation 2 with $\hat{\boldsymbol{s}}_0^* = 0$ and $\hat{\boldsymbol{s}}_{\hat{n}}^* = M$
7: Insert $\langle k, \hat{n}, \{\hat{\boldsymbol{s}}^*\}_{t=1}^{\hat{n}-1}\rangle$ to Q
8: **return** Q
9: **function** QUANCALC(dist $\langle k, \lambda\rangle$, table Q)
10: Lookup $\hat{\boldsymbol{s}}^* \leftarrow Q[k, n/2]$
11: **return** $\hat{\boldsymbol{s}}^* * \lambda$
12: **if** Q not initialized **then**
13: $Q \leftarrow$ INITQUANTABLE()
14: $\hat{\boldsymbol{s}}_{pos} \leftarrow$ QUANCALC(pos, Q)
15: $\hat{\boldsymbol{s}}_{neg} \leftarrow$ QUANCALC(neg, Q).negative().reverse()
16: **return** $\{\boldsymbol{x}.min, \hat{\boldsymbol{s}}_{neg}, 0, \hat{\boldsymbol{s}}_{pos}, \boldsymbol{x}.max\}$

Second, we apply the bucketing strategy where each tensor is sliced into buckets and each bucket is quantized individually. Such a bucketing technique is widely used to decrease the quantization variance (Seide et al., 2014; Alistarh et al., 2017). By default, we set bucket size as 4096, and we exclude small tensors (<10K values, typically, gradients of `bn` layers and batch-wise mean and stddev).

We finish this section by summarizing the complexity of TINYSCRIPT. To quantize a tensor, we need to (1) compute statistics of each bucket, (2) fit Weibull distributions, (3) calculate optimal points, and finally (4) perform the quantization. Fortunately, the time complexities of step 2 and 3 are both $O(1)$ for each bucket, and different buckets can be processed concurrently. The most time-consuming phases are step 1 and 4, which are also required in uniform quantization and can be highly parallelizable with SIMD instructions. As a result, the proposed non-uniform quantization algorithm shares the same complexity as that of uniform.



(a) Forward propagation. $\boldsymbol{u}, \boldsymbol{v}$ are forward units (activations).



(b) Backward propagation. $\boldsymbol{m}, \boldsymbol{n}$ are backward units.

Figure 4. An MLP model with ReLU as activation functions.

4. Analysis

In this section, we describe our analysis of the proposed non-uniform quantization algorithm.

4.1. Rationality of Weibull Prior

To comprehensively understand the rationality of the modeling method in Section 3.2, we investigate the distribution property of forward and backward units for an MLP model depicted in Figure 4. To begin with, we introduce the analysis of forward propagation in (Vladimirova et al., 2019).

Assumption 1. *For an MLP model depicted in Figure 4, we migrate the assumptions from (Vladimirova et al., 2019):*
(1) All weights are independent and have zero-mean normal distribution. Specifically, let the model have L layers and $\boldsymbol{W}^{(l)}$ have size $C_{l-1} \times C_l$, then we have $W_{i,j}^{(l)} \sim \mathcal{N}(0, {\sigma^{(l)}}^2)$ for $1 \leq l \leq L, 1 \leq i \leq C_{l-1}, 1 \leq j \leq C_l$;
(2) The input data are independent with model weights.

Theorem 2. [Theorem 3.1 of (Vladimirova et al., 2019)] *Consider an MLP depicted in Figure 4. If Assumption 1 holds, then for any $1 \leq l \leq L, 1 \leq i \leq C_l$, we have $u_i^{(l)}, v_i^{(l)} \sim \mathrm{subW}(l/2)$.*

Assumption 1 implies a Gaussian prior on weights, and indicates the data are unseen before, which holds at the very first training steps or in online settings. (Vladimirova et al., 2019) concludes that activations conform to a sub-Weibull distribution parameterized by depth l. In order to study the gradient distribution, we extend the analysis to backward propagation. However, since it is non-trivial to deduce on the backward units, we have to make another assumption:

Assumption 2. *We make the Gradient Independence assumptions following (Yang & Schoenholz, 2017):*
(1) the model weights used in backward propagation are different copies from those in forward propagation, but drawn i.i.d. from the same distributions;
(2) for $1 \leq l \leq L$, the backward units $\boldsymbol{m}^{(l)}, \boldsymbol{n}^{(l)}$ are independent from the forward units $\boldsymbol{u}^{(l)}, \boldsymbol{v}^{(l)}$.

Assumption 2 is implicitly used in (Schoenholz et al., 2016) and formally introduced by (Yang & Schoenholz, 2017). Although seeming unrealistic, Assumption 2 is actually a well-known statement in the mean field theory since it plays a crucial role in the derivation of back-propagation. Due to space limitation, we skip the discussion of correctness of Assumption 2 in this work and refer readers to other surveys (Yang, 2019). Note that $\boldsymbol{m}^{(l)}, \boldsymbol{n}^{(l)}$ are correlated with $\boldsymbol{u}^{(l+1)}, \boldsymbol{v}^{(l+1)}$ according to $\boldsymbol{W}^{(l)}$, so we do not assume they are independent. Next, we introduce Theorem 3.

Theorem 3. *Assume the MLP model in Theorem 2 satisfies Assumption 2, then for any $1 \leq l < L, 1 \leq i \leq C_l$, we have $m_i^{(l)}, n_i^{(l)} \sim \mathrm{subW}((L-l)/2)$.*

Theorem 3 shows that the backward units conform to a sub-Weibull distribution parameterized by $L - l$.

Since $\nabla \boldsymbol{W}^{(l)} = \boldsymbol{n}^{(l)}(\boldsymbol{v}^{(l-1)})^T$, it gives an intuitive idea that gradients are also Weibull-related variables. To be formal, we conclude with Theorem 4.

Theorem 4. *Considering the MLP model in Theorem 2, 3, for any $1 < l < L, 1 \leq i \leq C_{l-1}, 1 \leq j \leq C_l$, we have $\nabla \boldsymbol{W}_{ij}^{(l)} \sim \mathrm{subW}((L-1)/2)$.*

Remark 1. *Although the above discussion is based on MLP and requires assumptions which are ideal in practice, it provides a hint on the distribution of activations and gradients — for DNNs where L is large, the distribution is strongly related to a sub-Weibull distribution with a large θ (or a double-Weibull distribution with a small k). As stated in Section 3.2, a smaller k leads to more central tendency and long-tailed, which corresponds to the empirical distribution of activations and gradients in DNNs.*

4.2. Understanding Convergence against Variance

Till now, readers might suspect the significance of exhaustingly minimizing the quantization variance. In other words, it is non-intuitive how quantization variance affects convergence. To answer this question, we try to analyze the convergence of SGD with quantization on the following (non-convex) empirical risk minimization problem:

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} f(\boldsymbol{w}) = \frac{1}{N}\sum_i^N f_i(\boldsymbol{w}), \quad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \tilde{\boldsymbol{g}}_{i_t},$$

where α is step size, $\boldsymbol{g}_{i_t} = \nabla f_{i_t}(\boldsymbol{w}_t)$, and $\tilde{\boldsymbol{g}}_{i_t}$ represents the quantized version of $\boldsymbol{g}_{i_t}$. Our analysis is based on Assumption 3 and given in Theorem 5.

Assumption 3. *For $\forall i \in \{1, 2, ..., N\}$, $\boldsymbol{w}, \boldsymbol{w'} \in \mathbb{R}^D$, we make the assumptions below following (Allen-Zhu, 2017):*

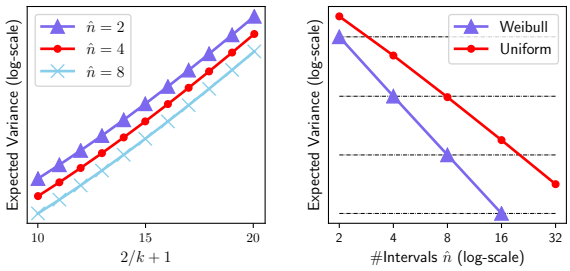


Figure 5. Expectation of quantization variance. Left: Expected variance of non-uniform quantization. We adjust the x-axis as $2/k+1$ to connect to L according to Theorem 4. Right: Comparison of expected variance for non-uniform and uniform quantization with $k=0.1$. The figures show that quantization variance almost exponentially increases w.r.t. $2/k+1$ and decreases w.r.t. $\log \hat{n}$.

(1. L_2-Lipschitz) $||\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w'})|| < L_2||\boldsymbol{w}-\boldsymbol{w'}||$;
(2. Bounded moment) $\mathbb{E}[||\nabla f_i(\boldsymbol{w})||] \leq \sigma_0$;
(3. Bounded variance) $\mathbb{E}[||\nabla f_i(\boldsymbol{w}) - \nabla f(\boldsymbol{w})||] \leq \sigma$;
(4. Existence of global minimum) $\exists f^*$ s.t. $f(\boldsymbol{w}) \geq f^*$.

Theorem 5. *Suppose we run SGD with quantization on an object satisfying Assumption 3 with constant step size $\alpha < 2/L_2$. Assume quantization satisfies $\mathbb{E}[||\tilde{\boldsymbol{x}} - \boldsymbol{x}||] \leq Q\mathbb{E}[||\boldsymbol{x}||]$ for any $x \in \mathbb{R}^D$. After T runs, select $\overline{\boldsymbol{w}}_T$ randomly from $\{\boldsymbol{w}_0, \boldsymbol{w}_1, ..., \boldsymbol{w}_{T-1}\}$. Then we have*

$$\mathbb{E}[||\nabla f(\overline{\boldsymbol{w}}_T)||^2] \leq \frac{2(f(\boldsymbol{w}_0)-f^*)}{(2\alpha-\alpha^2 L_2)T} + \frac{\alpha L_2(\sigma^2+Q^2\sigma_0^2)}{2-\alpha L_2}.$$

As T increases, the convergence of SGD with quantization is influenced by quantization variance Q, especially when the optimization is stuck in a sharp minima/minimum. For uniform quantization, Q can be bounded by $O(||\boldsymbol{x}||_\infty/\hat{n})$, which turns Theorem 5 into similar analysis in previous works (Alistarh et al., 2017; Bernstein et al., 2018). However, this only represents a worst-case scenario, which can hardly occur unless all values are close to one side of the intervals but unfortunately quantized to another side. Hence, we should consider the expectation of quantization variance, which gives a more general picture. We plot the expected variance in Figure 5 and summarize the discussion below.

Remark 2. *Upon Theorem 5, we further assume gradients follow double-Weibull distribution with $k = O(2/(L-1))$ inspired by Theorem 4. According to Figure 5, we conclude that $\mathbb{E}[Q] = O(q_1^L/q_2^{\log \hat{n}})$ for some constants $q_1 > 0, q_2 > 0$. By doing so, we have two important findings:*
(1) Since the expected variance exponentially increases w.r.t. L, a larger $\hat{n}$ (which turns into more bits) is needed to obtain comparable convergence for deeper networks;
(2) The proposed non-uniform quantization method can achieve similar variance as the uniform-based counterpart with half quantization levels.

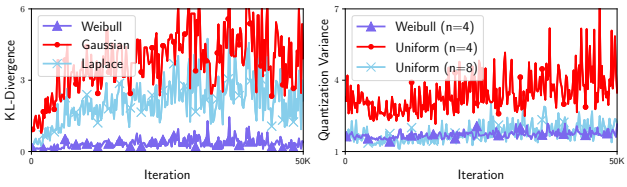


Figure 6. Measurements in different iterations (WRN34, Cifar10, gradients of a `conv` layer in the last residual block), the lower the better. Left: KL divergence between gradients and fitted distribution. Right: Related quantization variance $||\tilde{\boldsymbol{g}} - \boldsymbol{g}||^2/||\boldsymbol{g}||^2$.

5. Experiments

5.1. Experimental Setup

We conduct the experiments on a GPU server equipped with 4 Titan RTX GPUs. We focus on CNNs training since it is a well-studied task and models have non-trivial sizes [3]. Specifically, we consider the widely-used VGG and ResNet architectures (Simonyan & Zisserman, 2014; He et al., 2016; Zagoruyko & Komodakis, 2016). All experiments are conducted on Cifar10 and ImageNet (Krizhevsky et al., 2009; Deng et al., 2009).

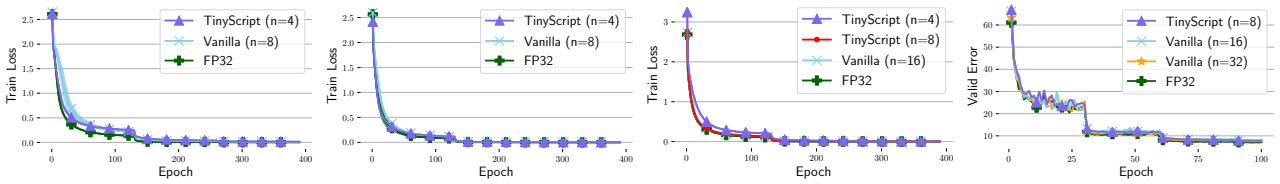
5.2. Effectiveness of Non-uniform Quantization

We first assess the effectiveness of the proposed non-uniform quantization algorithm over Cifar10 dataset and WideResNet34 (WRN34) model. Due to space limitation, we only take gradients of a `conv` layer in the last residual block as a representative since it has the largest size, whilst similar results are also observed on activations and other models.

Measurement of distribution To validate our analysis of distribution, we empirically compare gradients fitted with different distributions regarding KL divergence. Specifically, we fit Weibull distribution with Algorithm 1, and fit Gaussian and Laplace distributions via `scipy.stats` package. As shown in the left of Figure 6, throughout the training, fitting with Weibull distribution can always achieve the lowest KL divergence. In fact, Laplace distribution is a special case of double-Weibull when $k=1$, thus it does not have enough complexity to describe the distribution. Whilst Gaussian distribution has the poorest performance due to the stark difference to distribution.

Measurement of quantization The most critical issue we concern about is the quantization variance. As a result, we measure the quality of quantization by computing the related variance $||\tilde{\boldsymbol{g}} - \boldsymbol{g}||^2/||\boldsymbol{g}||^2$ in different iterations. Results are

[3] Since Section 4 targets at feed-forward networks, whether recurrent neural networks (RNNs) share similar distribution properties remains unknown. Therefore, we leave the discussion and evaluation on RNNs as our future work.



(a) Cifar10, VGG19 (b) Cifar10, WRN34 (c) Cifar10, ResNet50 (d) ImageNet, ResNet50

Figure 7. Convergence curves. For Cifar10, we plot the mean and stddev of training loss over 5 runs. For ImagetNet, we plot the top-5 validation error. We start from the quantization level $n = 4$ and double the level until comparable accuracy has been achieved.

Table 1. Best testing error on Cifar10 (in percentage). We report the mean and stddev of 5 runs. "-" indicates we skip the level since comparable accuracy has been achieved.

Model	Batch Size	Init. LR	TINYSCRIPT		Vanilla		FP32
			$n = 4$	$n = 8$	$n = 8$	$n = 16$	
VGG19	256	0.1	7.98±.20	-	7.81±.17	-	7.15±.03
WRN34	256	0.1	5.72±.12	-	5.78±.23	-	5.23±.25
ResNet50	128	0.05	6.45±.11	5.59±.18	diverge	5.61±.14	5.57±.07

Table 2. Best top-5 validation error on ImageNet (in percentage).

Method		Error
TINYSCRIPT	$n = 8$	7.74
Vanilla	$n = 16$	7.97
	$n = 32$	7.48
FP32		7.05

shown in the right of Figure 6. In general, our non-uniform approach consistently achieves lower variance compared to the uniform-based counterpart at the same quantization level $n = 4$ since it better adapts to the distribution and executes an optimal quantization. In contrast, the uniform-based approach requires doubling n ($n = 8$) to obtain a comparable variance, which is consistent with Figure 5.

5.3. Convergence Performance

We compare the convergence of TINYSCRIPT, uniform-based quantization (Vanilla) [4], and full precision (FP32) over Cifar10 and ImageNet datasets. Figure 7 presents the convergence curves and Table 1, 2 list the best performances.

Convergence on Cifar10 We first experiment on Cifar10 dataset with VGG19, WRN34, and ResNet50. We use SGD with momentum of 0.9 to train the models. In most cases, using a small learning rate is beneficial for the quantization methods since the error of each iteration can be restricted. However, a smaller learning rate requires more iterations to converge and even leads to more communication rounds in distributed training. As a result, to achieve a fair comparison, hyper-parameters are tuned for FP32 and applied to TINYSCRIPT and Vanilla directly. We make 5 runs for each method and report the mean and standard deviation. Overall, FP32 achieves the best model performance since it never suffers from quantization variance. Both TINYSCRIPT and Vanilla can converge to considerable accuracy (within 1% drop). However, Vanilla requires more quantization levels (larger n) to converge properly and even diverges in all cases when $n = 4$. This is unsurprising since TINYSCRIPT provides a lower quantization variance as described above, and hence gains better convergence. Furthermore, it is worthy to note that both TINYSCRIPT and Vanilla entail a higher performance drop on ResNet50 than VGG19 and WRN34. It matches our discussion in Section 4.2 that a deeper network would cause higher quantization variance and require more quantization levels.

Convergence on ImageNet We then compare the performance of each method on ImageNet dataset with ResNet50. Following (He et al., 2016), we set batch size as 256, initial learning rate as 0.1, and use SGD with momentum of 0.9. In general, FP32 still obtains the best model performance and the convergence curves of the quantization competitors are close to that of FP32. With a larger level $n = 32$, Vanilla incurs 0.43% drop in accuracy, whilst suffers from a 0.92% drop with $n = 16$ and diverges for smaller n. In contrast, the gap between FP32 and TINYSCRIPT is only 0.69% when $n = 8$. Furthermore, Vanilla diverges for both $n = 4$ and $n = 8$ owing to the high quantization variance, whilst the validation error of TINYSCRIPT is 10.5% when $n = 4$. The experiment on this large-scale dataset verifies the ability of TINYSCRIPT to achieve sound model performance with a more aggressive quantization.

Compression Finally, we compare the compression performance of TINYSCRIPT and Vanilla with the results in Figure 8. Although the existing uniform quantization conveys a possibility to train DNNs with a small level n so that memory consumption and communication cost can be reduced, our non-uniform quantization further decreases

[4] we implement Vanilla as a combination of the uniform quantization for gradients (Alistarh et al., 2017) and activations (Chakrabarti & Moseley, 2019). The techniques described in Section 3.4 are also applied to Vanilla to achieve fair comparison.

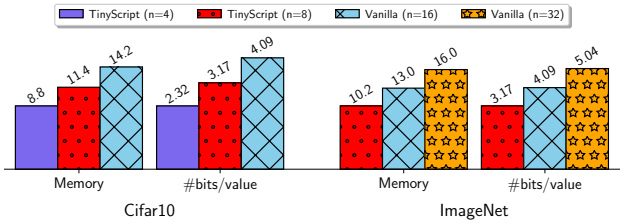


Figure 8. Compression performance on ResNet50 models. We show the memory consumed by activations in percentage of FP32 (we assume FP32 adopts re-computation) and the number of bits per value after quantization. Both metrics are better if lower.

n to one-half or even one-fourth. From the perspective of information theory, it requires at most $\log(n+1)$ bits to represent n levels. Thus, TINYSCRIPT enjoys approximately 0.8-1.8 bits saving compared to the uniform-based method.

6. Conclusion and Possible Future Works

This work proposes TINYSCRIPT, a non-uniform quantization framework for DNNs. TINYSCRIPT models the values with a family of Weibull distributions and leverages the distribution property to achieve minimum quantization variance. Empirical results show that TINYSCRIPT obtains lower quantization variance than the uniform-based mechanism, and therefore achieves a higher compression rate with model accuracy on par with full precision training.

Beyond this work, we wish to discover the strength of the proposed non-uniform quantization method in three directions. First, since all DNNs discussed in this work are feed-forward networks, it will be interesting to see whether our method fits RNNs. Second, whether the modeling on distribution works well on model weights is worth discussing. If yes, a model compression technique based on our method would be possible. Third, a dynamic control, e.g., choosing n w.r.t. k, may greatly improve the power of our method. We will leave the exploration of these topics as future works.

Acknowledgements

The work is supported by National Key R&D Program of China (No. 2018YFB1004403), National Natural Science Foundation of China (NSFC) (No. 61832001, 61702016, 61702015, U1936104), PKU-Baidu Fund 2019BD006, Beijing Academy of Artificial Intelligence (BAAI), PKU-Tencent joint research Lab, and Fundamental Research Funds for the Central Universities 2020RC25. CZ and the DS3Lab gratefully acknowledge the support from the Swiss National Science Foundation (Project Number 200021_184628), Swiss Data Science Center, Alibaba, eBay, Google Focused Research Awards, Oracle Labs, Zurich Insurance, Chinese Scholarship Council, and the Department of Computer Science at ETH Zurich.

References

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Allen-Zhu, Z. Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 89–97. JMLR. org, 2017.

Arun, K. and Govindan, V. A hybrid deep learning architecture for latent topic-based image retrieval. *Data Science and Engineering*, 3(2):166–195, 2018.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. Scalable methods for 8-bit training of neural networks. In *Advances in neural information processing systems*, pp. 5145–5153, 2018.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. Signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 559–568, 2018.

Cambier, L., Bhiwandiwalla, A., Gong, T., Nekuii, M., Elibol, O. H., and Tang, H. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. *arXiv preprint arXiv:2001.05674*, 2020.

Chakrabarti, A. and Moseley, B. Backprop with approximate activations for memory-efficient network training. In *Advances in Neural Information Processing Systems*, pp. 2426–2435, 2019.

Chen, S., Jian, Z., Huang, Y., Chen, Y., Zhou, Z., and Zheng, N. Autonomous driving: cognitive construction and situation understanding. *Science China Information Sciences*, 62(8):81101, 2019.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Drumond, M., Tao, L., Jaggi, M., and Falsafi, B. Training dnns with hybrid block floating point. In *Advances in Neural Information Processing Systems*, pp. 453–463, 2018.

Gharibshah, Z., Zhu, X., Hainline, A., and Conway, M. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering*, 5(1):12–26, 2020.

Greenwald, M., Khanna, S., et al. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jiang, J., Fu, F., Yang, T., and Cui, B. Sketchml: accelerating distributed machine learning with data sketches. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1269–1284. ACM, 2018.

Jiang, J., Fu, F., Yang, T., Shao, Y., and Cui, B. Skcompress: compressing sparse and nonuniform gradient in distributed machine learning. *The VLDB Journal*, pp. 1–28, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, K. and Li, G. Approximate query processing: What is new and where to go? *Data Science and Engineering*, 3 (4):379–397, 2018.

Mishchenko, K., Gorbunov, E., Takáč, M., and Richtárik, P. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., and Keckler, S. W. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–13. IEEE, 2016.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 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.

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

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

Sun, H., Shao, Y., Jiang, J., Cui, B., Lei, K., Xu, Y., and Wang, J. Sparse gradient compression for distributed sgd. In *International Conference on Database Systems for Advanced Applications*, pp. 139–155. Springer, 2019.

Vladimirova, M., Arbel, J., and Mesejo, P. Bayesian neural networks become heavier-tailed with depth. 2018.

Vladimirova, M., Verbeek, J., Mesejo, P., and Arbel, J. Understanding priors in bayesian neural networks at the unit level. In *International Conference on Machine Learning*, pp. 6458–6467, 2019.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pp. 9850–9861, 2018a.

Wang, L., Ye, J., Zhao, Y., Wu, W., Li, A., Song, S. L., Xu, Z., and Kraska, T. Superneurons: dynamic gpu memory management for training deep neural networks. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 41–53, 2018b.

Wang, N., Choi, J., Brand, D., Chen, C.-Y., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in neural information processing systems*, pp. 7675–7684, 2018c.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Wu, J., Huang, W., Huang, J., and Zhang, T. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pp. 5321–5329, 2018.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

Yang, G. and Schoenholz, S. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pp. 7103–7114, 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 4035–4043. JMLR. org, 2017.

Zhang, W., Jiang, J., Shao, Y., and Cui, B. Snapshot boosting: a fast ensemble framework for deep neural networks. *Science China Information Sciences*, 63(1):112102, 2020.