Finding the Task-Optimal Low-Bit Sub-Distribution in Deep Neural Networks

Runpei Dong[1][*] **Zhanhong Tan**[2][*] **Mengdi Wu**[2] **Linfeng Zhang**[2] **Kaisheng Ma**[2]

Abstract

Quantized neural networks typically require smaller memory footprints and lower computation complexity, which is crucial for efficient deployment. However, quantization inevitably leads to a distribution divergence from the original network, which generally degrades the performance. To tackle this issue, massive efforts have been made, but most existing approaches lack statistical considerations and depend on several manual configurations. In this paper, we present an adaptive-mapping quantization method to learn an optimal latent sub-distribution that is inherent within models and smoothly approximated with a concrete Gaussian Mixture (GM). In particular, the network weights are projected in compliance with the GM-approximated sub-distribution. This sub-distribution evolves along with the weight update in a co-tuning schema guided by the direct task-objective optimization. Sufficient experiments on image classification and object detection over various modern architectures demonstrate the effectiveness, generalization property, and transferability of the proposed method. Besides, an efficient deployment flow for the mobile CPU is developed, achieving up to 7.46× inference acceleration on an octa-core ARM CPU. Our codes have been publicly released at https://github.com/RunpeiDong/DGMS.

1. Introduction

Deep neural networks (DNNs) have become the *de-facto* method in plentiful applications, including computer vision (He et al., 2016; 2020), natural language processing (Devlin et al., 2019; Xu et al., 2020), speech recognition (Hinton et al., 2012), and robotics (Kim & Moon, 2016; Liu et al., 2016a). Although DNNs are pushing the

[*]Equal contribution [1]Xi'an Jiaotong University [2]Tsinghua University. Correspondence to: Kaisheng Ma <kaisheng@mail.tsinghua.edu.cn>.
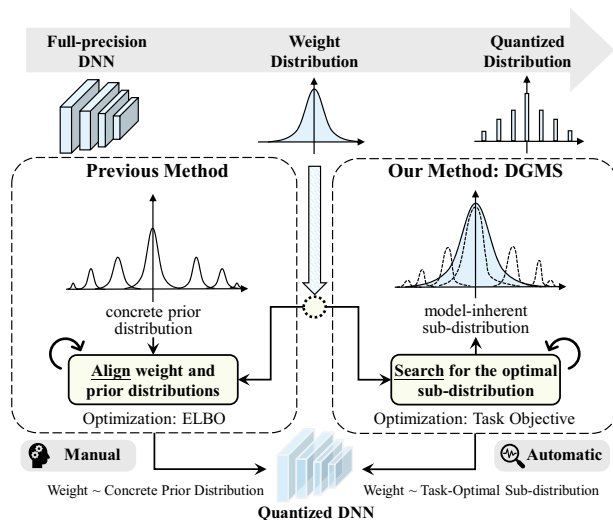
Figure 1: Comparison between the previous methods and our proposed DGMS. The training objective of the previous methods is ELBO while it is the task-objective only for DGMS, and the previous methods depend on manual configurations while our DGMS is fully automated.

limits of generalization performance across various areas, the growing memory and computation severely hinder the practical deployment. Abundant acceleration and compression methods have been proposed to tackle this issue, such as pruning (LeCun et al., 1989; Han et al., 2016), weight quantization (Fiesler et al., 1990; Rastegari et al., 2016), knowledge distillation (Hinton et al., 2015), efficient model design (Ma et al., 2018; Han et al., 2020; Tan et al., 2021), and even neural architecture search (Zoph et al., 2018).

Among the aforementioned schemes, weight quantization, as one of the mainstream techniques, aims to eliminate the representative redundancy via shortened bit-width. Generically, quantization can be represented as a projection $Q : \mathcal{W} \in \mathbb{R} \rightarrow \mathcal{Q} = \{q_0, q_1, \ldots, q_K\}$, where $\mathcal{W} \sim \mathcal{D}_{\mathcal{W}}$ is the real-valued preimage while $\mathcal{Q} \sim \mathcal{D}_{\mathcal{Q}}$ denotes the compressed and discrete representation. The categorical distribution $\mathcal{D}_{\mathcal{Q}}$ can be viewed as a factorized Dirac distribution for the expected precision, which inherently suffers a divergence from the preimage $\mathcal{D}_{\mathcal{W}}$. Consequently, the weight deviation after low-bit quantization generally leads to significant performance degradation.

To address the problem mentioned above, one typical solution is *straight through estimator* (STE) (Bengio et al., 2013) that simulates and encourages the network discretization via modifying the backpropagation (Wu et al., 2016; Hubara et al., 2017; Choi et al., 2018; Cai et al., 2017). However, most of these approaches depend on the fixed configurations (*e.g.*, centroids (Hubara et al., 2016; Zhou et al., 2017; Fiesler et al., 1990) and stepsize (Courbariaux et al., 2015; Gupta et al., 2015)), and it is tough to find the optimal solution. In addition, the global statistical information is not fully utilized, and thus the impact of numerical state transitions is omitted (Choi et al., 2018). Accordingly, a feasible solution is to manually set a concrete prior distribution $\mathcal{D}_{\mathcal{P}}$ that smoothly approximates the factorized Dirac posterior $\mathcal{D}_{\mathcal{Q}}$ through variational learning and reparametrization (Jang et al., 2017; Maddison et al., 2017; Louizos et al., 2017; Ullrich et al., 2017). As shown in Fig. 1, this method aims at minimizing the KL divergence $D_{\mathrm{KL}}(\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{Q}})$ during learning the task objective, which can be achieved directly by optimizing the *evidence lower bound* (ELBO) (Hinton & van Camp, 1993; Graves, 2011).

However, despite the fact that the differentiable learning with statistical information is adopted, it is still difficult to reach the ELBO optimum due to the large gradient variance and local optima trap. As has been pointed out by Carreira-Perpiñán & Idelbayev, there exists a gap between the prior and quantized distributions resulting from the separation of approximating the distribution and quantizing the weights in a two-step scheme. Besides, direct transfer of a manual-configured prior from one domain to another seems unappealing since it generally requires statistical alignment training.

In this paper, we introduce **D**ifferentiable **G**aussian **M**ixture Weight **S**haring (**DGMS**), a novel adaptive-mapping quantization method that statistically searches for the optimal low-bit model with a latent and representative sub-distribution. Compared to prior works, the superiority of DGMS benefits from three aspects: 1) DGMS relaxes the Dirac posterior into a concrete GM approximation to estimate the optimal sub-distribution that is model-inherent but not user-specified. 2) DGMS directly minimizes the task loss via weight and sub-distribution co-tuning for task-optimal quantization. 3) We further leverage the temperature-based softmax reparametrization (Hinton et al., 2015; Jang et al., 2017) to narrow the gap between training and inference.

In summary, the main contributions of this paper can be listed as follows:

- We propose a novel quantization method DGMS that statistically explores the optimal latent low-bit sub-distribution without handcrafted settings. Distributions and weights are trainable in a self-adaptive and end-to-end fashion.

- The promising transfer ability across four domains reveals the domain-invariant character, indicating the model-inherence of the found sub-distribution.

- Extensive experiments with several classical and lightweight DNNs demonstrate the remarkable compression performance and generalizability of DGMS on both image classification and object detection.

- We also evaluate DGMS-quantized low-bit models on the mobile CPU with an efficient deployment flow that maintains the parallelism and optimizes the cache access. The results show $1.66\times \sim 7.46\times$ speedup compared to the full-precision models.

2. Related Works

It has been acknowledged that DNNs are heavily over-parameterized with significant redundancy (Denil et al., 2013), and a wide variety of approaches have been proposed for the compact representation. The straightforward solution is to design a lightweight model with handcrafted efficient blocks, such as MobileNet (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019), ShuffleNet (Zhang et al., 2018b; Ma et al., 2018) and GhostNet (Han et al., 2020), or with Neural Architecture Search (NAS) (Zoph et al., 2018; Tan & Le, 2019; Xie et al., 2019). Another attractive direction is to eliminate the inherent redundancy in existing architectures by using Knowledge Distillation (KD) (Hinton et al., 2015; Bucila et al., 2006; Zhang et al., 2021), pruning (Hagiwara, 1993; Han et al., 2015; 2016), or quantization (Fiesler et al., 1990; Rastegari et al., 2016).

In particular, quantization is widely recognized to be an efficient manner for DNN deployment on commercial products (Liao et al., 2019; Bannon et al., 2019; Jiao et al., 2020; Lin et al., 2020) profiting from the natural regularity for processing. Quantization techniques can be basically categorized into two groups: 1) discontinuous-mapping quantization based on rounding operation, and 2) continuous-mapping quantization based on adaptive reparameterization.

Discontinuous-Mapping Quantization One line of works employs stochastic rounding (Gupta et al., 2015; Hubara et al., 2017; Wu et al., 2018), where the mapping decisions are stochastically made among quantization intervals. Particularly, RQ (Louizos et al., 2019) is similar to our method, but it employs this stochastic rounding operation for fixed quantization grids. The distribution assumption is made upon the input signal noise while we directly model the weights. In parallel, the other line of works focuses on deterministic rounding to snap the full-precision weights to the closest quantized centroids towards binary (Courbariaux et al., 2015), ternary (Marchesi et al., 1993), or power-of-two quantization (Tang & Kwan, 1993). Starting

with the typical pioneering studies (Courbariaux & Bengio, 2016; Courbariaux et al., 2015), many researchers adopt STE (Bengio et al., 2013) to tackle the non-differentiability issue caused by the rounding operation (Hubara et al., 2017; Rastegari et al., 2016). However, the pseudo-gradients may lead to an inaccurate optimization direction, and statistical information is often ignored. Meanwhile, many methods (Zhang et al., 2018a; Qu et al., 2020; Xu et al., 2018) have been proposed to explore Multi-Bit Quantization (MBQ) (Lin et al., 2017). Specifically, DMBQ (Zhao et al., 2021) also focuses on distribution following (Banner et al., 2019), which employs a fixed Laplace prior to reformulate the quantization error for optimization. Recently, KD (Mishra & Marr, 2018; Polino et al., 2018) and Reinforcement Learning (RL) (Elthakeb et al., 2020; Wang et al., 2020) are introduced for better efficiency-accuracy trade-off, generally requiring indirect efforts and expertise.

Continuous-Mapping Quantization This solution is able to utilize global statistical information. Several works manually define a concrete prior distribution for approximating the quantized categorical distribution via variational learning (Louizos et al., 2017; Ullrich et al., 2017; Shayer et al., 2018; Soudry et al., 2014) or Markov Chain Monte Carlo (MCMC) (Roth & Pernkopf, 2020). Though allowing for the differentiable learning, the training is confronted with the convergence difficulty or large memory footprint for MCMC sampling (Neal, 1993). Also, similar to STE-based methods, a heuristic manual prior is needed. Our method is closely related to this continuous-mapping scheme but different from previous works. Our proposed DGMS automatically searches for a model-inherent sub-distribution without manual configuration, and the optimization directly targets the task-objective (see the illustration in Fig. 1 and discussions in Appendix B.2).

To avoid the discussed issues, the primary objective of this paper is to peer inside the model-inherent statistical information for quantization in an end-to-end differentiable manner. Our inspiration comes from the recently proposed "Lottery Ticket Hypothesis" in the pruning literature (Frankle & Carbin, 2019; Frankle et al., 2020), which proves that there exists an optimal *sub-network* for a randomly initialized DNN. For quantization, we aim to explore whether there exists an optimal latent *sub-distribution* for a given full-precision DNN that can be tuned to match the performance of the original network.

3. Methodology

In this section, we first present a reformulation of conventional network quantization and then introduce our proposed DGMS quantization method for the optimal sub-distribution searching. Finally, we describe an efficient mobile deployment flow named Q-SIMD for DGMS.

3.1. Preliminaries: Quantization Reformulation with Auxiliary Indicator

As stated in Sec. 1, a network quantizer Q maps the continuous preimage $x \in \mathbb{R}^M$ to a discrete finite set $\mathcal{Q}$. For example, a symmetric uniform quantizer Q_U maps given inputs in a linear fashion:

$$Q_U(\mathcal{W}) = \text{sign}(\mathcal{W}) \odot \begin{cases} \Delta \left\lfloor \frac{|\mathcal{W}|}{\Delta} + \frac{1}{2} \right\rfloor, & |\mathcal{W}| \leq \max(\mathcal{Q}_U) \\ \max(\mathcal{Q}_U), & |\mathcal{W}| > \max(\mathcal{Q}_U) \end{cases} \tag{1}$$

where $\odot$ denotes the Hadamard product, Δ is the predetermined stepsize and $\mathcal{Q}_U = \{0, \pm k\Delta, k = 1, \ldots, K\}$ is the uniform centroids.

The common problem is that this discontinuous-mapping scheme may limit the representational capacity of quantization, and the nearly zero derivatives of Eqn. (1) are generally useless. Hence, we employ an adaptive quantization set $\mathcal{Q}_A = \{\mu_0, \mu_1, \ldots, \mu_K\}$ where $\mu_0 = 0$ and $\mu_1, \ldots, \mu_K$ are adaptive values. Besides, an auxiliary indicator $\mathcal{I}^Q : \mathbb{R}^M \to \{0, 1\}^{K \times M}$ is introduced for $\mathcal{W}$ given any condition $\mathcal{A}$ (i.e., $\mathcal{I}^Q(\mathcal{W}) = 1$ if $\mathcal{A}$ holds and zero otherwise), which has been widely used for pruning (Mallya et al., 2018; Lee et al., 2019):

$$\mathcal{I}_k^Q(\mathcal{W}) = \begin{cases} 1, & \text{if } \mathcal{W} \text{ is quantized to } \mu_k \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$
$$\text{for } k = 0, 1, \ldots K.$$

The quantization procedure can be reformulated as follows:

$$Q_A(\mathcal{W}) = \sum_{k=0}^{K} \mu_k \odot \mathcal{I}_k^Q(\mathcal{W}). \tag{3}$$

Here, $Q_A(\mathcal{W}) \in \mathcal{Q}_A$ is the quantized low-bit representation which is defined as a one-hot linear combination of $\mathcal{Q}_A$ in Eqn. (3). With the foregoing reformulations, we further regard the auxiliary indicator as a probability-based decision variable to relax the quantization operations and serve for the end-to-end differentiable training.

3.2. DGMS: Differentiable Gaussian Mixture Weight Sharing

First of all, DGMS aims to search for a model-inherent sub-distribution that is defined as follows:

Definition 1. Given the preimage $\mathcal{W} \sim \mathcal{D}_{\mathcal{W}}$ and quantized data $\mathcal{Q} \sim \mathcal{D}_{\mathcal{Q}}$, the sub-distribution $\mathcal{D}_{\mathcal{S}}$ is defined as an estimation for $\mathcal{D}_{\mathcal{W}}$ (i.e., $\mathcal{D}_{\mathcal{S}} \approx \mathcal{D}_{\mathcal{W}}$, where $\approx$ denotes approximate equivalence), and under a parameter limitation τ, $\mathcal{D}_{\mathcal{S}}$ is approximately equivalent to $\mathcal{D}_{\mathcal{Q}}$ (i.e., $\lim_{\tau \to 0} \mathcal{D}_{\mathcal{S}} \approx \mathcal{D}_{\mathcal{Q}}$).

As defined above, the sub-distribution serves as a distributional bridge between the full-precision and quantized

representations, which is controlled by a hyper-parameter τ introduced later. In order to better understand Definition 1, we provide the theoretical discussions in Appendix B.1.

3.2.1. GAUSSIAN MIXTURE SUB-DISTRIBUTION APPROXIMATION

There are a few approaches to generate the sub-distribution $\mathcal{D}_S$ satisfying the first condition $\mathcal{D}_S \cong \mathcal{D}_W$ in Definition 1, *e.g.*, analyzing the statistical property with maximum likelihood estimation (MLE) based on a parameterized approximation. Empirically, the pretrained weights distribute like a bell-shaped Gaussian or Laplacian (Han et al., 2016; Lin et al., 2016). Thus, following Nowlan & Hinton, we assume a mixture of $K + 1$ uni-modal Gaussian components as a smooth sub-distribution approximation for the full-precision network. This Gaussian Mixture (GM) approximation GM_ϑ parameterized by $\vartheta = \{\pi_k, \mu_k, \gamma_k\}_{k=0}^K$ (π_k: prior attributing probability, μ_k: mean, γ_k: standard deviation) is capable of forming a strong representation of arbitrary-shaped densities (Reynolds, 2009), revealing the statistical knowledge within the model. For efficiency and simplicity, we initialize the GM approximation with the k-means algorithm, which can be viewed as a special case of expectation-maximization (EM) algorithm (Dempster et al., 1977).

3.2.2. STATISTICAL GUIDED WEIGHT SHARING

The magnitude-based weight salience has been early explored (Han et al., 2015; 2016), demonstrating that the larger weights tend to outweigh those with small magnitudes. Based on this observation, we naturally divide the weights into several regions using the parameterized GM. Each region is associated with a Gaussian component, where the mean value serves as a region salience for the weight sharing, *i.e.* the adaptive quantization set $\mathcal{Q}_A = \{\mu_0, \mu_1, \ldots, \mu_K\}$ where $\mu_0 = 0$. Similar to Eqn. (2), here we define a region decision indicator $\mathcal{I}^S : \mathbb{R}^M \to \{0, 1\}^{K \times M}$ for all $K + 1$ regions based on the GM sub-distribution approximation. Then, for each data point $\mathbf{w}_j$, the quantized low-bit representation $\Psi(\mathbf{w}_j; \vartheta) \sim \mathcal{D}_\mathcal{Q}$ is formulated as:

$$\mathcal{I}_k^S(\mathbf{w}_j; \vartheta) = \begin{cases} 1, & \text{if } \arg\max_i \varphi(\mathbf{w}_j, i) = k \\ 0, & \text{otherwise} \end{cases}, \quad (4)$$
$$\text{for } k = 0, 1, \ldots, K;$$

$$\Psi(\mathbf{w}_j; \vartheta) = \sum_{k=0}^K \mu_k \odot \mathcal{I}_k^S(\mathbf{w}_j; \vartheta), \quad (5)$$

where $\varphi(\mathbf{w}_j, k) \in [0, 1]$ denotes the posterior probability $p(\mathbf{w}_j \in \text{region k}|\vartheta_k)$ of $\mathbf{w}_j$ within the given region k (parameterized by $\vartheta_k = \{\pi_k, \mu_k, \gamma_k\}$). This can be written as:

$$\varphi(\mathbf{w}_j, k) = \frac{\exp\left(\pi_k \mathcal{N}(\mathbf{w}_j | \mu_k, \gamma_k^2)\right)}{\sum_{i=0}^K \exp\left(\pi_i \mathcal{N}(\mathbf{w}_j | \mu_i, \gamma_i^2)\right)}. \quad (6)$$

Eqn. (6) is the soft-normalized region confidence according to the GM sub-distribution. It can also be regarded as the nearest clustering based on a weighted radial basis function (RBF) kernel (Broomhead & Lowe, 1988) that calculates a soft symmetry distance between weights $\mathbf{w}_j$ and the corresponding region salience.

3.2.3. DIFFERENTIABLE INDICATOR

However, recalling Eqn. (4), the hard sampling operation $\arg\max$ makes it completely non-differentiable, making it difficult to optimize with gradient descent. To tackle this issue, there exist many techniques such as Gumbel softmax (Jang et al., 2017). In this paper, we simply adopt the temperature-based softmax (Hinton et al., 2015; Maddison et al., 2017), which shifts the region confidence closer to a one-hot encoding with low temperatures, bridging the gap between the indicators during training and inference. With the introduced technique, the distributional sampling procedure is then differentiable and can be a well estimated region confidence prediction for $\mathcal{I}_k^S(\mathbf{w}_j; \vartheta)$:

$$\phi_k(\mathbf{w}_j; \vartheta, \tau) = \frac{\exp\left(\varphi(\mathbf{w}_j, k)/\tau\right)}{\sum_{i=0}^K \exp\left(\varphi(\mathbf{w}_j, i)/\tau\right)}, \quad (7)$$
$$\text{for } k = 0, 1, \ldots, K,$$

where τ is set as a learnable temperature parameter that adjusts the discretization estimation level. Further, the compressed representation $\Phi_k(\mathbf{w}_j; \vartheta, \tau) \sim \mathcal{D}_S$ can be reformulated as follows:

$$\Phi(\mathbf{w}_j; \vartheta, \tau) = \sum_{k=0}^K \mu_k \odot \phi_k(\mathbf{w}_j; \vartheta, \tau). \quad (8)$$

3.2.4. TRAINING AND INFERENCE

The proposed DGMS quantization can be summarized in Algorithm 1. Note that DGMS can be deployed along different granularities, such as channel-wise or filter-wise. In this paper, layer-wise DGMS is explored since it is faster to train.

3.3. Q-SIMD: Efficient Deployment on the Mobile CPU

DGMS not only reduces the memory footprint but also speedups the inference for deployment. To evaluate the model acceleration benefiting from DGMS, we develop an efficient flow to deploy our quantized 4-bit networks on the mobile CPU. The deployment faces two challenges: 1) to fetch real weights with 4-bit indices while maintaining the parallelism (i.e., SIMD), and 2) to optimize the shifted cache bottleneck of activations after the weight compression.

Given that the smallest numeric data type in current ARM-v8 is BYTE, simply employing the 4-bit indices wastes

---

Algorithm 1 Model compression using Differentiable Gaussian Mixture Weight Sharing.

Input: Training dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ like ImageNet and DNN $\mathcal{F}$ of L layers with full-precision weights $\mathcal{W} = \{\mathbf{w}^\ell\}_{\ell=1}^L$, GM component number $K + 1$ and initial temperature $\{\tau^\ell\}_{\ell=1}^L$.

1: **Initialization**
2: **for** $\ell \leftarrow 1$ **to** L **do**
3:    $\mathcal{R} \leftarrow \{\mathbf{w}^\ell | \mathbf{w}^\ell \in \text{region } k\}_{k=0}^K$;   ▷ `initial`
      `region generation with MLE`
4:    $\min_{k=0}^K (|\hat{\mu}_k|) \leftarrow 0, \quad \vartheta^\ell \leftarrow$
     $\left\{\hat{\mu}_k, \hat{\pi}_k \leftarrow \frac{|\mathcal{R}_k|}{|\mathbf{w}^\ell|}, \hat{\gamma}_k \leftarrow \sqrt{\frac{\sum_{j=1}^{|\mathbf{w}^\ell|}(\mathbf{w}_j^\ell - \hat{\mu}_k)^2}{|\mathbf{w}^\ell| - 1}}\right\}_{k=0}^K$;
5: **end for**
6: $\Theta \leftarrow \{\vartheta^\ell, \tau^\ell\}_{\ell=1}^L$;
7: **Training**
8: **while** not converged **do**
9:    **for** $\ell \leftarrow 1$ **to** L **do**
10:      **for** $k \leftarrow 0$ **to** K **do**
11:        $\phi_k(\mathbf{w}^\ell; \vartheta^\ell, \tau^\ell) \leftarrow \frac{\exp\left(\varphi(\mathbf{w}^\ell, k)/\tau^\ell\right)}{\sum_{i=0}^K \exp\left(\varphi(\mathbf{w}^\ell, i)/\tau^\ell\right)}$,
       Eqn. (6) and Eqn. (7);
12:      **end for**
13:      $\Phi(\mathbf{w}^\ell; \vartheta^\ell, \tau^\ell) \leftarrow \sum_{k=0}^K \mu_k^\ell \odot \phi_k(\mathbf{w}^\ell; \vartheta^\ell, \tau^\ell)$,
     Eqn. (8);
14:    **end for**
15:    Evaluate with task loss, *e.g.*, Cross Entropy
     $\mathcal{L}_{\text{CE}}\Big(\mathcal{F}(\mathcal{X}; \Phi(\mathcal{W}; \Theta)), \mathcal{Y}\Big)$;
16:    Backpropagation and update $\{\mathcal{W}, \Theta\}$ with the stochastic gradient descent;
17: **end while**
18: **Inference**
19: $\widetilde{\mathcal{W}} \leftarrow \Psi(\mathcal{W}; \Theta)$, Eqn. (4) and Eqn. (5);
20: Output prediction $\widehat{\mathcal{Y}}$ with quantized neural network: $\widehat{\mathcal{Y}} = \mathcal{F}(\mathcal{X}; \widetilde{\mathcal{W}})$.

---

the high half-byte for the 8-bit alignment. Therefore, we propose an 8-bit associated index that can access multiple weights at a time. For example, we use a 256-entry codebook to represent a 4-bit index set, where each entry contains a couple of weights, (W_X, W_Y). The associated index fully utilizes 8 bits, and naturally, it is conducive to loading data in parallel for SIMD operations. Even though the codebook size is squared, the overhead can be ignored compared to the whole index tensor. Besides, after weight quantization, the memory bottleneck has turned to be the activation access. To reduce cache misses, we further place the activation-related iterations to the outer loops so that activations can be kept local in the L1 cache without data reloading. More details can be found in Appendix A.

4. Experiments

4.1. Experimental Setup

4.1.1. DATASETS AND MODELS

To demonstrate the effectiveness and generalization ability of DGMS, we evaluate our method on image classification and object detection. Classification experiments are conducted on CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). PASCAL VOC (Everingham et al., 2015) dataset is used as the detection benchmark. We use VOC2007 plus VOC2012 `trainval` for training and evaluate on VOC2007 `test`. We principally select a series of lightweight models to achieve further compression, which is more significant for an advanced quantization technique at present. For CIFAR-10, we evaluate DGMS on ResNet-20, ResNet-32, ResNet-56 (He et al., 2016) and VGG-Small (Zhang et al., 2018a) (a small-sized variant of VGG-Net (Simonyan & Zisserman, 2015)). For ImageNet, we use ResNet-18 and ResNet-50 for evaluations (He et al., 2016). Besides, we evaluate DGMS on lightweight architectures including MobileNetV2 (Sandler et al., 2018) and two NAS-based MnasNet-A1 (Tan et al., 2019) and ProxylessNAS-Mobile (Cai et al., 2019). The experiments on PASCAL VOC are performed on SSDLite, a publicly available lite version of SSD (Liu et al., 2016b). Here, we use SSDLite-MBV2 and SSDLite-MBV3 with MobileNetV2 (Sandler et al., 2018) and MobileNetV3 (Howard et al., 2019) as the backbone models, respectively.

4.1.2. DEPLOYMENT

We extend the TVM framework (Chen et al., 2018) to support the Q-SIMD flow introduced in Sec. 3.3. Our evaluation is performed on the octa-core ARM CPU in Qualcomm 888 (Samsung S21). We conduct various image classification and object detection models quantized by DGMS to evaluate their runtime using the Android TVM RPC tool. The baseline results are obtained under full-precision models using the primitive optimal TVM scheduling.

4.2. Image Classification Results

4.2.1. CIFAR-10 RESULTS

As shown in Tab. 1, we compare the proposed DGMS on CIFAR-10 to LQNets (Zhang et al., 2018a) which is trained with Quantization Error Minimization (QEM), and TTQ (Zhu et al., 2017) which is trained with task-objective optimization only (see discussions in Sec. B.2). From the experimental results, we observe that DGMS consistently achieves a promising model sparsity at the same bit-width with a more lightweight representation while maintaining the performance. For example, the 2-bit ResNet-20 architecture that is extremely tiny can still achieve 44.44% parameter

Table 1: Comparison across different quantization methods on CIFAR-10. #Params: the number of non-zero model parameters, Bits: weights quantization bit-width, TOO: task-objective optimization only. Compared methods are TTQ (Zhu et al., 2017) and LQNets (Zhang et al., 2018a).

Model	Method	TOO	Bits	#Params	Top-1 Acc.
ResNet-20	Our FP32	N/A	FP32	0.27M	93.00%
	LQNets	×	3	0.27M	92.00%
	Ours	✓	3	**0.20M**	**92.84%**
	LQNets	×	2	0.27M	91.80%
	Ours	✓	2	**0.15M**	**92.13%**
ResNet-32	Our FP32	N/A	FP32	0.46M	94.27%
	TTQ	✓	2	0.46M	92.37%
	Ours	✓	2	**0.27M**	**92.97%**
ResNet-56	Our FP32	N/A	FP32	0.85M	94.61%
	TTQ	✓	2	0.85M	93.56%
	Ours	✓	2	**0.44M**	**93.72%**
VGG-Small	Our FP32	N/A	FP32	4.66M	94.53%
	LQNets	×	3	4.66M	93.80%
	Ours	✓	3	**3.12M**	**94.46%**
	LQNets	×	2	4.66M	93.80%
	Ours	✓	2	**2.24M**	**94.36%**

reduction. Besides, when quantizing models under a lower-precision bit-width setting, a higher model-inherent sparsity is found by DGMS, which is derived from the decreasing number of GM components (see Sec. 4.5.1).

4.2.2. IMAGENET RESULTS

In Tab. 2, we show the classification performance of compressed models on the ImageNet benchmark, compared to several strong baselines including LQNets (Zhang et al., 2018a), HAQ (Wang et al., 2020), TQT (Jain et al., 2020), AutoQ (Lou et al., 2019), UNIQ (Baskin et al., 2021), HAWQV3 (Yao et al., 2021) and joint pruning-quantization methods including BB (van Baalen et al., 2020), CLIP-Q (Tung & Mori, 2020) and ANNC (Yang et al., 2020). Note that we adopt full-precision models from torchvision 0.7.0 and torch hub. Since this work focuses on weights quantization, the weights are quantized with DGMS, and activations are quantized in a post-training fashion following Li et al.. Tab. 2 shows that for the 4-bit quantization setting, our DGMS-quantized ResNet-18 and ResNet-50 achieve even higher classification accuracy than the full-precision models. Even though lightweight NAS-based models are challenging for compression, the efficient MnasNet-A1 and ProxylessNAS-Mobile quantized by our DGMS lead to a negligible accuracy loss. As a result, DGMS presents a competitive or better classification accuracy with consistently decent CRs in comparison to both quantization-only and joint pruning-quantization model compression methods.

Table 2: Comparison across different quantization-only and joint quantization-pruning methods on ImageNet. P+Q: joint pruning-quantization methods. Compared methods are LQNets (Zhang et al., 2018a), AutoQ (Lou et al., 2019), HAQ (Wang et al., 2020), TQT (Jain et al., 2020), BB (van Baalen et al., 2020), CLIP-Q (Tung & Mori, 2020), ANNC (Yang et al., 2020), UNIQ (Baskin et al., 2021) and HAWQV3 (Yao et al., 2021).

Model	Method	P+Q	W/A	Top-1 Acc.	Δ Acc.
ResNet-18	Our FP32	×	FP32	69.76%	N/A
	LQNets	×	4/32	70.00%	+0.40%
	Ours	×	4/32	**70.25%**	**+0.49%**
	LQNets	×	3/32	69.30%	-0.30%
	Ours	×	3/32	**69.57%**	**-0.19%**
	BB[♭]	✓	4/8	69.60%	-0.08%
	UNIQ	×	4/8	67.02%	-2.58%
	HAWQV3	×	4/8	**70.22%**	-1.25%
	Ours	×	4/8	70.12%	**+0.46%**
	AutoQ	×	4/4	67.30%	-2.60%
	HAWQV3	×	4/4	68.45%	-3.02%
	Ours	×	4/4	**68.95%**	**-0.81%**
ResNet-50	Our FP32	×	FP32	76.15%	N/A
	UNIQ	×	4/32	75.09%	-0.93%
	HAQ	×	4/32 mixed	76.14%	-0.01%
	Ours	×	4/32	**76.28%**	**+0.13%**
	CLIP-Q[♮]	✓	3/32	73.70%	**+0.60%**
	HAQ	×	3/32 mixed	75.30%	-0.85%
	Ours	×	3/32	**75.91%**	-0.24%
	HAQ	×	2/32 mixed	70.63%	-5.52%
	Ours	×	2/32	**72.88%**	**-3.27%**
	UNIQ	×	4/8	74.37%	-1.65%
	TQT	×	4/8	74.40%	-1.00%
	HAWQV3	×	4/8 mixed	75.39%	-2.33%
	Ours	×	4/8	**76.22%**	**+0.07%**
	AutoQ	×	4/4	72.43%	-2.37%
	HAWQV3	×	4/4	74.24%	-3.48%
	Ours	×	4/4	**75.05%**	**-1.10%**
MnasNet-A1	Our FP32	×	FP32	73.51%	N/A
	ANNC[♯]	✓	4/32	71.80%	-1.66%
	Ours	×	4/32	**72.18%**	**-1.33%**
	Ours	×	4/8	**72.09%**	**-1.42%**
ProxylessNAS -Mobile	Our FP32	×	FP32	74.59%	N/A
	ANNC[♯]	✓	4/32	72.46%	-2.13%
	Ours	×	4/32	**73.85%**	**-0.74%**
	Ours	×	4/8	**73.24%**	**-1.35%**

[♭] Result estimated from the figure plot reported in the original paper. The weight CR of BB is 8.93× while for DGMS the CR is 9.41×.

[♮] The weight CR of CLIP-Q is 31.81× while DGMS yields CR of 14.75× and 31.37× for 3-bit and 2-bit ResNet50, respectively.

[♯] The MnasNet CR is 17.10× with ANNC compared to 12.40× with DGMS; the ProxylessNAS CR is 16.80× with ANNC compared to 9.02× with DGMS.

Table 3: Compression results for SSDLite detectors on PASCAL VOC2007. For each model, the first and the second row show the full model performance (%) and our compressed detector, respectively.

Model	#Params	Bits	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog	horse	mbike	person	plant	sheep	sofa	table	train	tv
SSDLite -MBV2	3.39M	32	68.6	69.7	78.2	63.4	54.8	35.6	78.8	74.4	82.0	53.8	61.9	78.5	82.2	80.6	71.8	42.8	62.6	78.4	73.7	83.3	65.5
	2.68M	4	67.9	69.4	78.5	63.6	54.6	34.2	77.6	73.1	82.4	53.1	61.1	76.3	81.3	81.3	70.7	41.0	62.3	78.8	72.6	81.1	65.0
SSDLite -MBV3	2.39M	32	67.2	66.0	79.7	58.7	54.9	36.9	79.3	73.0	83.3	51.1	62.8	77.7	81.5	77.1	70.0	39.0	58.9	74.6	72.0	82.2	64.8
	1.26M	4	65.7	65.5	77.9	55.5	54.5	34.3	77.7	72.1	83.8	48.7	59.6	76.2	80.2	75.8	69.2	37.2	57.0	72.3	72.5	80.8	62.5

4.3. Object Detection Results

To demonstrate the generalization ability of our proposed DGMS, we quantize the lightweight detectors SSDLite-MBV2 and SSDLite-MBV3 to 4-bit models on PASCAL VOC benchmark. Note that our method can be deployed on these detectors directly with no additional modification efforts. As displayed in Tab. 3, we observe $10.12\times$ and $15.17\times$ compression rates respectively for SSD-MBV2 and -MBV3 while retaining the detection accuracy. For some objects like *motor-bike* (mbike), our quantized detectors can achieve more accurate perception, and we argue that the degradation comes from the lack of training samples and the intrinsic detection difficulty for tiny objects.

4.4. Deployment Results on the Mobile CPU

Tab. 4 compares the inference performance of the Q-SIMD flow with primitive TVM scheduling on the mobile CPU. It is observed that a 4.34-to-7.46$\times$ speedup is achieved for ResNet-18 and ResNet-50, respectively. Lightweight DNNs have scant room because separable convolutions are widely used in models with less weight reuse probability and reduced computation. Despite this, a 1.66-to-1.74$\times$ speedup is achieved in three lightweight models.

Table 4: Deployment results on the mobile CPU of Qualcomn 888 for the ImageNet and PASCAL VOC2007 runtime (ms). Baseline: full-precision models based on primitive TVM (Chen et al., 2018). DGMS: 4-bit quantized models based on Q-SIMD deployment.

Model	ResNet-18	ResNet-50	MnasNet-A1	ProxylessNAS	SSDLite$^\diamond$
Baseline	67.09 ms	148.56 ms	18.27 ms	22.17 ms	43.95 ms
DGMS	**8.99 ms**	**34.26 ms**	**10.49 ms**	**13.34 ms**	**25.60 ms**
Speedup	7.46$\times$	4.34$\times$	1.74$\times$	1.66$\times$	1.72$\times$

$^\diamond$ SSDLite-MBV2 with MobileNetV2 as the backbone model.

4.5. Ablation Study

4.5.1. GM COMPONENT NUMBER

Fig. 2 plots the Top-1 accuracy (%) of 4-bit quantized ResNet-18 on ImageNet. The corresponding compression rates with varying numbers of GM components (*i.e.*, varying

K). The figure exhibits that the compression rates slightly decline when the accuracy boosts for large values of K. It indicates that GM with fewer components is more compact with higher sparsity, which spontaneously results from the lower degree of representation freedom.
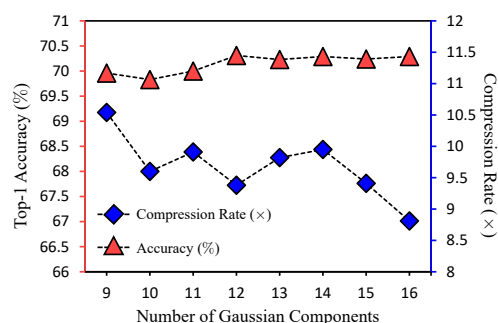

Figure 2: ResNet-18 ImageNet performance and CR with varying numbers of GM components.

5. Discussions

5.1. Gaussian Mixture Evolution

For DGMS, the quantization mapping is adaptively learned with the GM evolution during training. In order to further understand this weight-distribution co-tuning schema, as shown in Fig. 3, we visualize the weight redistribution process of 2-bit quantization (*i.e.*, $K = 3$) with DGMS and the learning curve of ResNet-20 on CIFAR-10. Fig. 3(a) shows the pretrained weight distribution with the initial GM densities plotted. Fig. 3(b) and (c) respectively illustrate the distributions of weights after DGMS and quantization. We can observe that the proposed DGMS is capable of tuning the GM according to the continuous distribution. Besides, the model-inherent statistical information is modeled, and the weights are accordingly redistributed. Furthermore, the weights projected by DGMS serve as a fine hard-quantized estimation, which manages to narrow the gap between the original and compressed representations. Fig. 3(d) plots the validation performance and the weight deviation measured by mean square error (MSE) of the uncompressed and quantized weights. Note that no such quantization error is used
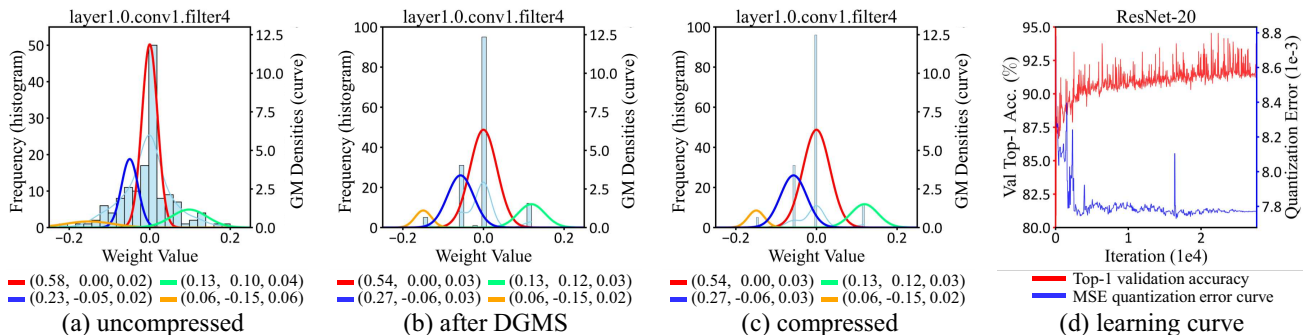
(0.58, 0.00, 0.02) (0.13, 0.10, 0.04) (0.23, -0.05, 0.02) (0.06, -0.15, 0.06)	(0.54, 0.00, 0.03) (0.13, 0.12, 0.03) (0.27, -0.06, 0.03) (0.06, -0.15, 0.02)	(0.54, 0.00, 0.03) (0.13, 0.12, 0.03) (0.27, -0.06, 0.03) (0.06, -0.15, 0.02)	Top-1 validation accuracy MSE quantization error curve
(a) uncompressed	(b) after DGMS	(c) compressed	(d) learning curve

Figure 3: (a)-(c) Weight distribution and the Gaussian densities of GM. The legend (x, y, z) represents ($\pi = x, \mu = y, \gamma = z$). (d) Top-1 validation accuracy (%) and MSE quantization error curve.

Table 5: Top-1 accuracy of 4-bit quantized ResNet-18 across domains. DGMS without (w/o) transfer is the vanilla DGMS on the Target domain using GM initialization method introduced in Algorithm 1, and with (w/) transfer denotes GM initialization with sub-distribution found in the Source domain (Source→Target domain sub-distribution transfer).

Target		**Im**a**ge**Net			**CUB**200-2011			Stanford **Cars**			FGVC **Air**craft		
FP32 Full-Model		69.76%			78.68%			86.58%			80.77%		
4-bit DGMS (w/o transfer)		70.25%			77.90%			86.39%			80.41%		
4-bit DGMS (w/ transfer)	Source	CUB	Cars	Air	Img	Cars	Air	Img	CUB	Air	Img	CUB	Cars
	ZERO-SHOT	34.69%	62.31%	35.09%	73.53%	74.29%	66.44%	82.28%	81.46%	71.75%	77.46%	74.97%	77.31%
	ONE-EPOCH	68.37%	69.13%	68.80%	77.70%	77.54%	77.50%	85.70%	85.84%	85.79%	79.90%	79.87%	80.14%

for DGMS training, but it can be seen that the quantization error spontaneously declines with accuracy boosting.

5.2. Domain Invariance

In Tab. 5, we evaluate the transfer ability with 4-bit ResNet-18 across ImageNet, CUB200-2011 (Welinder et al., 2010), Stanford Cars (Krause et al., 2013), and FGVC Aircraft (Maji et al., 2013). We directly transfer the parameterized sub-distribution from the source domain to perform DGMS quantization and report the Top-1 accuracy (%) without any or with only *one-epoch* tuning. It is noteworthy that the zero epoch results are *zero-shot* quantization results on the target dataset via domain transfer. From the table, we can observe that except for the large-scale ImageNet dataset, all the models achieve competitive performances compared to full-precision models with zero training. This is because the sub-distribution learning has converged enough for the three medium-scale datasets, but it is not adapted to ImageNet with a reparable gap. Therefore, after only one epoch of fine-tuning with DGMS, the models are able to match the performance of uncompressed or vanilla DGMS-quantized models (trained for 60 epochs). This demonstrates the domain-invariant character of the sub-distribution, which is a model-inherent representation with superior generalization performance and training efficiency. Beyond that, this also provides a novel angle orthogonal to existing methods for zero-shot model quantization , *i.e.*, through domain generalization of adaptive quantization parameters.

6. Conclusions

In this paper, we introduce a novel quantization method named DGMS to automatically find a latent task-optimal low-bit sub-distribution, forming a distributional bridge between full-precision and quantized representations. Weights are projected based on the GM-parameterized sub-distribution approximation, which evolves with weights in a joint fashion by directly optimizing the task-objective. As a result, DGMS adaptively quantizes the DNNs with a found task-optimal sub-distribution and achieves negligible performance loss with high compression rates. Extensive experiments on both image classification and detection on over-parameterized and lightweight DNNs have been conducted, demonstrating a competitive compression performance compared to other state-of-the-art methods and the transferability of the obtained model-inherent sub-distributions across different domains. To date, the formulation and optimization direction remain open problems in the quantization community. Among numerous existing solutions, we propose to view the optimal quantization configuration searching in a different way as to find the task-optimal sub-distribution. We also provide a great potential to solve zero-shot quantization through domain generalization, and we hope our work could spur future related researches. In the future, we would like to explore DGMS with a bimodal distribution (*e.g.*, arcsine distribution) for binary DNNs, and it would be intriguing to solve the task-optimal sub-distribution searching problem with techniques like reinforcement learning.

References

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, pp. 7948–7956, 2019.

Bannon, P., Venkataramanan, G., Sarma, D. D., and Talpes, E. Computer and redundancy solution for the full self-driving computer. In *IEEE Hot Chips 31 Symposium (HCS)*, pp. 1–22. IEEE, 2019.

Baskin, C., Liss, N., Schwartz, E., Zheltonozhskii, E., Giryes, R., Bronstein, A. M., and Mendelson, A. UNIQ: uniform noise injection for non-uniform quantization of neural networks. *ACM Trans. Comput. Syst.*, 37(1-4): 4:1–4:15, 2021.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

Broomhead, D. and Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 535–541. ACM, 2006.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Cai, Z., He, X., Sun, J., and Vasconcelos, N. Deep learning with low precision by half-wave gaussian quantization. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 5406–5414. IEEE Computer Society, 2017.

Carreira-Perpiñán, M. Á. and Idelbayev, Y. Model compression as constrained optimization, with application to neural nets. part II: quantization. *CoRR*, abs/1707.04319, 2017.

Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E. Q., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., Guestrin, C., and Krishnamurthy, A. TVM: an automated end-to-end optimizing compiler for deep learning. In *USENIX Symp. Oper. Syst. Des. Implement. (OSDI)*, pp. 578–594. USENIX Association, 2018.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I., Srinivasan, V., and Gopalakrishnan, K. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

Courbariaux, M. and Bengio, Y. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

Courbariaux, M., Bengio, Y., and David, J. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 3123–3131, 2015.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *J. R. Stat. Soc.: Ser. B (Methodol.)*, 39(1):1–22, 1977.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 248–255. IEEE Computer Society, 2009.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. Predicting parameters in deep learning. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 2148–2156, 2013.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., and Esmaeilzadeh, H. Releq : A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE Micro*, 40(5):37–45, 2020.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. M., and Zisserman, A. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis. (IJCV)*, 88(2):303–338, 2010.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis. (IJCV)*, 111(1):98–136, January 2015.

Fiesler, E., Choudry, A., and Caulfield, H. J. Weight discretization paradigm for optical neural networks. In *Optical interconnections and networks*, volume 1281, pp. 164–173. International Society for Optics and Photonics, 1990.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 119 of

Proceedings of Machine Learning Research, pp. 3259–3269. PMLR, 2020.

Graves, A. Practical variational inference for neural networks. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 2348–2356, 2011.

Grünwald, P. D. *The minimum description length principle*. MIT press, 2007.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1737–1746. JMLR.org, 2015.

Hagiwara, M. Removal of hidden units and weights for back propagation networks. pp. 351–354, 1993.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. Ghostnet: More features from cheap operations. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 1577–1586. Computer Vision Foundation / IEEE, 2020.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Adv. Neural Inform. Process. Syst. (NIPS)*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 770–778. IEEE Computer Society, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 42(2):386–397, 2020.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, pp. 5–13. ACM, 1993.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Howard, A., Pang, R., Adam, H., Le, Q. V., Sandler, M., Chen, B., Wang, W., Chen, L., Tan, M., Chu, G., Vasudevan, V., and Zhu, Y. Searching for mobilenetv3. In *Int. Conf. Comput. Vis. (ICCV)*, pp. 1314–1324. IEEE, 2019.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 4107–4115, 2016.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18:187:1–187:30, 2017.

Jain, S. R., Gural, A., Wu, M., and Dick, C. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. In Dhillon, I. S., Papailiopoulos, D. S., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems (MLSys)*, volume 2, pp. 112–128, 2020.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2017.

Jiao, Y., Han, L., Jin, R., Su, Y.-J., Ho, C., Yin, L., Li, Y., Chen, L., Chen, Z., Liu, L., He, Z., Yan, Y., He, J., Mao, J., Zai, X., Wu, X., Zhou, Y., Gu, M., Zhu, G., Zhong, R., Lee, W., Chen, P., Chen, Y., Li, W., Xiao, D., Yan, Q., Zhuang, M., Chen, J., Tian, Y., Lin, Y., Wu, W., Li, H., and Dou, Z. A 12nm programmable convolution-efficient neural-processing-unit chip achieving 825tops. In *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020.

Kim, Y. and Moon, T. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE Geosci. Remote. Sens. Lett.*, 13(1):8–12, 2016.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *Int. Conf. Comput. Vis. Worksh. (ICCV Workshop)*, pp. 554–561. IEEE Computer Society, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 598–605. Morgan Kaufmann, 1989.

Lee, N., Ajanthan, T., and Torr, P. H. S. Snip: single-shot network pruning based on connection sensitivity. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., and Gu, S. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2021.

Liao, H., Tu, J., Xia, J., and Zhou, X. Davinci: A scalable architecture for neural network computing. In *IEEE Hot Chips 31 Symposium (HCS)*, pp. 1–44. IEEE, 2019.

Lin, C.-H., Cheng, C.-C., Tsai, Y.-M., Hung, S.-J., Kuo, Y.-T., Wang, P. H., Tsung, P.-K., Hsu, J.-Y., Lai, W.-C., Liu, C.-H., Wang, S.-Y., Kuo, C.-H., Chang, C.-Y., Lee, M.-H., Lin, T.-Y., and Chen, C.-C. A 3.4-to-13.3tops/w 3.6tops dual-core deep-learning accelerator for versatile ai applications in 7nm 5g smartphone soc. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 134–136. IEEE, 2020.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2849–2858. JMLR.org, 2016.

Lin, X., Zhao, C., and Pan, W. Towards accurate binary convolutional neural network. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 345–353, 2017.

Liu, F., Shen, C., Lin, G., and Reid, I. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 38(10):2024–2039, 2016a.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. SSD: single shot multibox detector. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 9905 of *Lecture Notes in Computer Science*, pp. 21–37. Springer, 2016b.

Lou, Q., Guo, F., Kim, M., Liu, L., and Jiang, L. Autoq: Automated kernel-wise neural network quantization. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. In *Adv. Neural Inform. Process. Syst. (NIPS)*, pp. 3288–3298, 2017.

Louizos, C., Reisser, M., Blankevoort, T., Gavves, E., and Welling, M. Relaxed quantization for discretized neural networks. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Ma, N., Zhang, X., Zheng, H., and Sun, J. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 11218 of *Lecture Notes in Computer Science*, pp. 122–138. Springer, 2018.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2017.

Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. Technical report, University of Oxford, 2013.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 11208 of *Lecture Notes in Computer Science*, pp. 72–88. Springer, 2018.

Marchesi, M., Orlandi, G., Piazza, F., and Uncini, A. Fast neural networks without multipliers. *IEEE Trans. Neural Networks*, 4(1), 1993.

Mishra, A. K. and Marr, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2018.

Neal, R. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Dept. Comput. Sci., Univ. Toronto, Toronto, Ontario, Rep. no. CRG-TR-92–1, 1993.

Nowlan, S. J. and Hinton, G. E. Simplifying neural networks by soft weight-sharing. *Neural Comput.*, 4(4):473–493, 1992.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, pp. 8024–8035, 2019.

Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2018.

Qu, Z., Zhou, Z., Cheng, Y., and Thiele, L. Adaptive loss-aware quantization for multi-bit networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 7985–7994. Computer Vision Foundation / IEEE, 2020.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Eur. Conf. Comput. Vis. (ECCV)*, pp. 525–542. Springer, 2016.

Reynolds, D. A. Gaussian mixture models. *Encyclopedia of biometrics*, 741:659–663, 2009.

Rissanen, J. Modeling by shortest data description. *Autom.*, 14(5):465–471, 1978.

Rissanen, J. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986. ISSN 00905364.

Roth, W. and Pernkopf, F. Bayesian neural networks with weight sharing using dirichlet processes. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 42(1):246–252, 2020.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018.

Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

Shayer, O., Levi, D., and Fetaya, E. Learning discrete weights using the local reparameterization trick. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2015.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

Soudry, D., Hubara, I., and Meir, R. Expectation back-propagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Adv. Neural Inform. Process. Syst. (NIPS)*, volume 1, pp. 2, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 6105–6114. PMLR, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 2820–2828. Computer Vision Foundation / IEEE, 2019.

Tan, S. H., Dong, R., and Ma, K. Multi-glimpse network: A robust and efficient classification architecture based on recurrent downsampled attention. In *Brit. Mach. Vis. Conf. (BMVC)*. BMVA Press, 2021.

Tang, C. Z. and Kwan, H. K. Multilayer feedforward neural networks with single powers-of-two weights. *IEEE Trans. Signal Process.*, 41(8), 1993.

Tung, F. and Mori, G. Deep neural network compression by in-parallel pruning-quantization. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 42(3), 2020.

Ullrich, K., Meeds, E., and Welling, M. Soft weight-sharing for neural network compression. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2017.

van Baalen, M., Louizos, C., Nagel, M., Amjad, R. A., Wang, Y., Blankevoort, T., and Welling, M. Bayesian bits: Unifying quantization and pruning. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2020.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Hardware-centric automl for mixed-precision quantization. *Int. J. Comput. Vis. (IJCV)*, 128(8):2035–2048, 2020.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. Quantized convolutional neural networks for mobile devices. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 4820–4828. IEEE Computer Society, 2016.

Wu, S., Li, G., Chen, F., and Shi, L. Training and inference with integers in deep neural networks. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2018.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2019.

Xu, C., Yao, J., Lin, Z., Ou, W., Cao, Y., Wang, Z., and Zha, H. Alternating multi-bit quantization for recurrent neural networks. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2018.

Xu, C., Zhou, W., Ge, T., Wei, F., and Zhou, M. Bert-of-theseus: Compressing BERT by progressive module replacing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869. Association for Computational Linguistics, 2020.

Yang, H., Gui, S., Zhu, Y., and Liu, J. Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In

IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR), pp. 2175–2185. Computer Vision Foundation / IEEE, 2020.

Yao, Z., Dong, Z., Zheng, Z., Gholami, A., Yu, J., Tan, E., Wang, L., Huang, Q., Wang, Y., Mahoney, M., et al. Hawq-v3: Dyadic neural network quantization. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11875–11886. PMLR, 2021.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Eur. Conf. Comput. Vis. (ECCV)*, pp. 373–390. Springer, 2018a.

Zhang, L., Bao, C., and Ma, K. Self-distillation: Towards efficient and compact neural networks. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 2021.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 6848–6856. Computer Vision Foundation / IEEE Computer Society, 2018b.

Zhao, S., Yue, T., and Hu, X. Distribution-aware adaptive multi-bit quantization. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 9281–9290. Computer Vision Foundation / IEEE, 2021.

Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2017.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2017.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 8697–8710. Computer Vision Foundation / IEEE Computer Society, 2018.
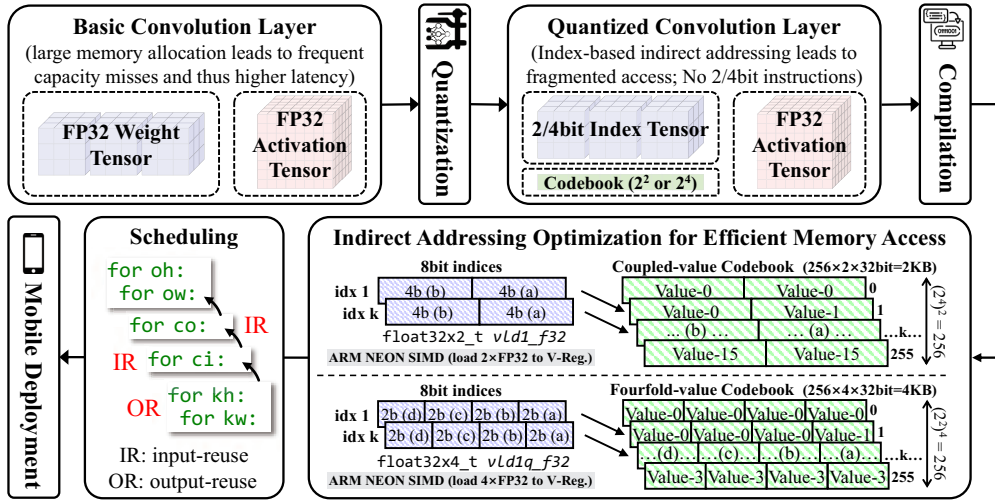
A. Details of Q-SIMD

Figure 4: Overview of Q-SIMD compilation framework for weight sharing quantization on the mobile SIMD CPU. The scheduling also employs loop tiling and unrolling (not shown in the figure). `kw`, `kh`: width and height of kernel. `ci`, `co`: input and output channel. `ow`, `oh`: width and height of output feature map.

Fig. 4 shows the detailed illustration of our Q-SIMD flow. DGMS firstly quantizes a full-precision model to a low-bit model of 2-bit or 4-bit weights. Because the real values of weights are still FP32, the compact model is stored as a low-bit index tensor with the corresponding codebook. Naturally, when executing the multiply-add operations, the real weights are loaded through indirect addressing with indices. However, the current ISA does not support 4-bit or 2-bit operations and requires at least 8-bit alignment. Additionally, the indirect addressing is irregular and thus damages the SIMD operations.

To address this problem mentioned above, we introduce an extended codebook with coupled or fourfold values for 4-bit and 2-bit quantization, respectively. In this way, the number of codebook entries is increased to 256, and then the index bit-width is changed to 8-bit. With the ARM NEON SIMD instructions (*e.g.*, `float32x2_t vld1_f32` and `float32x4_t vld1q_f32`), we can avoid bit wasting for low-bit alignment and maintain the parallel data loading for SIMD operations.

Furthermore, given that we mainly focus on weight quantization, the bottleneck of memory access turns to be activations. Therefore, we reduce the activation reloading via output-reuse (OR) and input-reuse (IR) scheduling. For the inner-loops, we keep the outputs local in the L1 cache to avoid reloading the partial sums. For the outer-loops, we put the `oh` and `ow` dimensions to reuse inputs across different output channels.

The aforementioned operation is registered as a customized operator in TVM (Chen et al., 2018). With the Android TVM RPC tool, the quantized models can be conveniently deployed on the ARM CPU and evaluated for inference time.

B. Further Discussions

B.1. Sub-Distribution

In this section, we discuss about sub-distribution introduced in Definition 1, which is essential and serves for the moral of this paper.

B.1.1. STATISTICAL APPROXIMATION

The concept "approximate equivalence", denoted as $\approxeq$, is statistical and it means the divergence between the estimated and the real data distribution does not exceed a small error ϵ.

B.1.2. FULL-PRECISION DISTRIBUTION ESTIMATION

The full-precision distribution estimation is the first condition in Definition 1, *i.e.*, $\mathcal{D}_\mathcal{S} \cong \mathcal{D}_\mathcal{W}$. As stated before, the GM sub-distribution is initialized by the MLE method, and MLE can model the statistical feature of data samples based on a parameterized approximation (Dempster et al., 1977). Hence, the parameterized GM is approximately equivalent to the data's real distribution, which can not be precisely measured but can be statistically estimated (Reynolds, 2009).

B.1.3. CATEGORICAL DISTRIBUTION APPROACHING

The quantization distribution approaching is the second condition in Definition 1, *i.e.*, $\lim_{\tau \to 0} \mathcal{D}_\mathcal{S} \cong \mathcal{D}_\mathcal{Q}$. This means after hard quantization, the weights are re-distributed to discrete values according to the categorical distribution $\mathcal{D}_\mathcal{Q}$ that originates from the concrete GM. Formally, given full-precision weight $\mathbf{w} \sim \mathcal{D}_\mathcal{W}$, $\Phi(\mathbf{w}) \sim \mathcal{D}_\mathcal{S}$ as the statistically shared weights according to the GM model and $\Psi(\mathbf{w}) \sim \mathcal{D}_\mathcal{Q}$ as the quantized value. Assuming that $k = \arg\max_i \varphi(\mathbf{w}, i)$, then we have $\lim_{\tau \to 0} \phi_k(\mathbf{w}) = 1$ and $\lim_{\tau \to 0} \phi_j(\mathbf{w}) = 0|_{j \neq k}$, hence it can be easily proved that $\lim_{\tau \to 0} \Phi(\mathbf{w}) = \Psi(\mathbf{w})$ (notations taken from Sec. 3.2).

B.2. MDL View on DGMS

Supposing that we have the training dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ and neural network $\mathcal{F}$ with weights $\mathcal{W}$, DGMS directly optimizes the objective with gradient descent:

$$\min_{\mathcal{W}, \vartheta, \tau} \mathcal{L}^E = -\log p\Big(\mathcal{Y}|\mathcal{X}, \Phi(\mathcal{W}; \vartheta, \tau)\Big), \tag{9}$$

where the Gaussian Mixture is parameterized by $\vartheta = \{\pi_k, \mu_k, \gamma_k\}_{k=1}^K$ and weights are smoothly quantized by the differentiable indicator defined in Eqn. (8).

From the minimum description length (MDL) view in information theory (Shannon, 1948; Grünwald, 2007), $\mathcal{L}^E$ is the lower bound of the information cost expectation to fit the data (Rissanen, 1978; 1986). In contrast, the previous works (Jang et al., 2017; Maddison et al., 2017; Ullrich et al., 2017; Roth & Pernkopf, 2020) optimize the *evidence lower bound* (ELBO) to train DNNs with smoothed stochastic weights from an estimated categorical prior distribution (*e.g.* Gaussian Mixture). The objective can be written with the error cost $\mathcal{L}^E$ plus the complexity cost $\mathcal{L}^C$, and for DGMS this can be given as follows:

$$\min_{\mathcal{W}, \vartheta, \tau} \underbrace{-\log p\Big(\mathcal{Y}|\mathcal{X}, \Phi(\mathcal{W}; \vartheta)\Big)}_{\mathcal{L}^E} + \underbrace{-\log p\Big(\Phi(\mathcal{W}, \vartheta, \tau)\Big) + \mathcal{H}\Big(q_\vartheta(\mathcal{W})\Big)}_{\mathcal{L}^C}, \tag{10}$$

where $\mathcal{L}^C$ amounts to the Kullback-Leibler divergence between the prior distribution and the Bayesian posterior distribution $q_\vartheta(\mathcal{W})$, and $\mathcal{H}$ is the entropy term which can be approximately reduced to a constant (Ullrich et al., 2017). In this paper, the complexity cost $\mathcal{L}^C$ is dropped while the weights and the parameterized GM co-adapt for redistribution.

B.3. Hardware Deployment

B.3.1. LIMITATIONS OF CODEBOOK-BASED FULLY-ADAPTIVE QUANTIZATION

The direct quantization (*e.g.*, 4-bit uniform quantization) can avoid the indirect addressing. However, this kind of quantization is not accuracy-friendly. In this work, even though the codebook takes up the memory footprint, the overhead is fairly small because our quantization granularity is layer-wise. Thanks to the small memory footprint and the cache locality, the codebook can be held in the L1-cache, whose access latency is fairly low. Besides, the codebook-based quantization may lead to irregular memory access from indirect addressing. To solve this problem, we introduce the index-pair to access data in parallel (see Q-SIMD deployment paradigm in Sec. A).

B.3.2. DETAILED MEMORY FOOTPRINT OF CODEBOOKS

The memory footprint of parameters in a quantized layer (*e.g.*, 4-bit) is calculated by: #params$\times 4$ (bit). The codebook overhead is calculated by: $2^4 \times 32$ (bit). For example, in the 4-bit quantized ResNet-18, ResNet-50, MnasNet-A1, SSDLite-MBV2, and ProxylessNAS, the overhead ratios of codebook are: 0.02%, 0.03%, 0.2%, 0.3%, and 0.3%.

C. Details of Experiments

C.1. Datasets

CIFAR-10   The CIFAR-10 dataset (Krizhevsky, 2009) consists of 60,000 32×32 tiny images for 10 classes object recognition, with 6,000 images per class.

ImageNet   The ImageNet (ILSVRC 2012) dataset (Deng et al., 2009) is a challenging large-scale dataset for image classification, containing 1.3M training samples and 50K test images with 1,000 object classes for recognition.

PASCAL VOC   The PASCAL VOC dataset (Everingham et al., 2015; 2010) is widely used as both semantic segmentation and object detection benchmarks. For object detection, VOC2007 `trainval` and VOC2012 `trainval` respectively contain 5,011 images and 11,540 images for training (16,551 images in all) and VOC07 `test` contains 4,952 images for evaluation. PASCAL VOC involves 21 classes including a background class for detection and segmentation.

CUB200-2011   The CUB200-2011 dataset (Welinder et al., 2010) is a medium-scale dataset for the fine-grained classification task, which contains 11,788 images in all for 200 bird species (5,994 samples for training and 5,794 images for validation).

Stanford Cars   The Stanford Cars dataset (Krause et al., 2013) contains a total of 16,185 car images of 196 categories, of which 8,144 samples are used for training and 8,041 images are used for validation.

FGVC Aircraft   The FGVC Aircraft dataset (Maji et al., 2013) consists of 10,000 images for 100 aircraft variants, the training set involves 6,667 samples and validation set involves 3,333 images.

C.2. Implementation Details

Optimization   We set the batch size as 128 on CIFAR-10 and 256 on ImageNet for all the models. SGD with 0.9 momentum and 5×10^{-4} weight-decay is used during training. All the models are trained for 350 epochs and 60 epochs respectively on CIFAR-10 and ImageNet. The max learning rate is set to 2×10^{-5} (1×10^{-5} for 2-bit ResNet-50) using one-cycle scheduler (Smith & Topin, 2019) and the initial temperature τ in Eqn. (7) is 0.01 for all the experiments. The training configurations for transferability experiments on the medium-scale datasets (CUB200-2011, Stanford Cars, and FGVC Aircraft) are the same as ImageNet. For PASCAL VOC detection, the batch size is 32 with 1×10^{-5} learning rate and we employ the cosine scheduler. The detectors are trained for only 10 epochs. With regard to the data augmentation, we simply adopt random crop and horizontal flip for image classification and the same augmentation operations as that in public source[1] for object detection.

Implementation   The experiments are implemented with PyTorch 1.6 (Paszke et al., 2019) on PH402 SKU 200 with 32G memory GPU devices. All the layers except the first and the last are quantized, which has become a common practice for model quantization. Similarly, for detectors, we perform the quantization on all layers except for the first layer of the model and the last layer of the classification head and regression head.

Table 6: Ablation study on initialization. The results are Top-1 classification accuracy (%) on CIFAR-10.

Model	Bits	Std. Init.	Empirical Initialization		
			$\gamma = 0.010$	$\gamma = 0.005$	$\gamma = 0.001$
ResNet-20	2	89.26%	92.13%	90.88%	91.30%
	3	92.16%	92.84%	92.54%	92.34%
VGG-Small	2	92.55%	94.36%	94.40%	94.41%
	3	94.30%	94.46%	94.56%	94.52%

---

[1] https://github.com/qfgaohao/pytorch-ssd.

D. Further Ablation Study

D.0.1. INITIALIZATION

For the GM initialization, we adopt k-means which can be simply and efficiently implemented on GPU devices[2]. In the CIFAR-10 experiments, we test two types of initialization for the parameter γ, including standard deviation initialization and empirical initialization. As a result, we empirically find that the calculation of standard deviation γ in Algorithm 1 may not be optimal but is simple and effective compared to the empirical initialization (see Tab. 6).

D.0.2. TEMPERATURE

As has been stated in Sec. 3.2, the temperature hyper-parameter τ controls the discretization estimation level. And when τ approaches zero, the estimated representation is approximately equivalent to the quantized one. This hyper-parameter can be set as a fixed or learnable parameter with a predefined initialized value. In Tab. 7, we test 3-bit VGG-Small on CIFAR-10 with different initializations. One can observe that a relatively lower temperature is more recommended to bridge the quantization gap, and a fixed temperature can achieve competitive performances compared to an adaptively learned one.

Table 7: Ablation study on temperature hyper-parameter τ. The results are Top-1 classification accuracy (%) with 3-bit VGG-Small on CIFAR-10.

Temperature Type	Initialization Value		
	τ=0.100	τ=0.010	τ=0.001
Fixed	93.57%	94.56%	94.27%
Learned	93.54%	94.64%	94.04%

E. Broader Impact

The rapid development of deep learning brings huge convenience and advanced productivity to our society, together with a higher daily cost (*e.g.*, carbon footprint). The proposed DGMS is a novel quantization method to address this issue, and it can be utilized to compress a variety of DNNs. In real life, this technique provides an eco-friendly and energy-efficient deployment solution for deep learning models, profiting from lower power consumption and less CO_2 emission. Though DGMS facilitates the AI development on edge devices like mobile phones, if without legal restraint, it will potentially lead to the infringement of image rights and personal privacy.

---

[2]https://github.com/subhadarship/kmeans_pytorch.