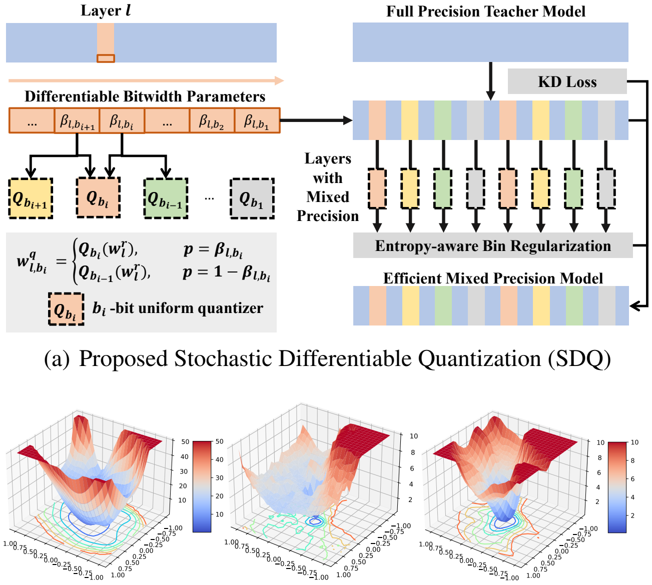
# SDQ: Stochastic Differentiable Quantization with Mixed Precision

Xijie Huang [1] Zhiqiang Shen [1 2 3] Shichao Li [1] Zechun Liu [4] Xianghong Hu [1 5] Jeffry Wicaksana [1]
Eric Xing [6 2] Kwang-Ting Cheng [1]

## Abstract

In order to deploy deep models in a computationally efficient manner, model quantization approaches have been frequently used. In addition, as new hardware that supports mixed bitwidth arithmetic operations, recent research on mixed precision quantization (MPQ) begins to fully leverage the capacity of representation by searching optimized bitwidths for different layers and modules in a network. However, previous studies mainly search the MPQ strategy in a costly scheme using reinforcement learning, neural architecture search, etc., or simply utilize partial prior knowledge for bitwidth assignment, which might be biased and sub-optimal. In this work, we present a novel **S**tochastic **D**ifferentiable **Q**uantization **(SDQ)** method that can automatically learn the MPQ strategy in a more flexible and globally-optimized space with smoother gradient approximation. Particularly, Differentiable Bitwidth Parameters (DBPs) are employed as the probability factors in stochastic quantization between adjacent bitwidth choices. After the optimal MPQ strategy is acquired, we further train our network with entropy-aware bin regularization and knowledge distillation. We extensively evaluate our method for several networks on different hardware (GPUs and FPGA) and datasets. SDQ outperforms all state-of-the-art mixed or single precision quantization with a lower bitwidth and is even better than the full-precision counterparts across various ResNet and MobileNet families, demonstrating the effectiveness and superiority of our method.


Layer $l$
Full Precision Teacher Model
KD Loss
Differentiable Bitwidth Parameters
... $\beta_{l,b_{i+1}}$ $\beta_{l,b_i}$ ... $\beta_{l,b_2}$ $\beta_{l,b_1}$
$Q_{b_{i+1}}$ $Q_{b_i}$ $Q_{b_{i-1}}$ ... $Q_{b_1}$
Layers with Mixed Precision
$w^q_{l,b_i} = \begin{cases} Q_{b_i}(w^r_l), & p = \beta_{l,b_i} \\ Q_{b_{i-1}}(w^r_l), & p = 1 - \beta_{l,b_i} \end{cases}$
$Q_{b_i}$ $b_i$-bit uniform quantizer
Entropy-aware Bin Regularization
Efficient Mixed Precision Model

(a) Proposed Stochastic Differentiable Quantization (SDQ)

(b) Full Precision (c) Interpolation (d) Stochastic

*Figure 1.* 1(a) demonstrates how our SDQ generates optimal MPQ strategy and train a MPQ network effectively. 1(b), 1(c), 1(d) compares the underlying loss optimization landscape (Li et al., 2018) of a full precision model, MPQ ResNet20 with linear interpolation (Yang & Jin, 2021; Nikolić et al., 2020), and MPQ ResNet20 with stochastic quantization.

## 1. Introduction

Deep learning models such as Convolutional Neural Networks (CNNs) have demonstrated outstanding performance on various tasks, such as visual recognition (Krizhevsky et al., 2012; He et al., 2016; Tan & Le, 2019). Their latency, energy consumption, and model size have become the most significant factors for consideration for the deployment of these deep learning models. In recent years, researchers have studied the design, training, and inferencing techniques of deep learning models for greater computation efficiency, including compact network design and search (Howard et al., 2017; Pham et al., 2018; Guo et al., 2020), knowledge distillation (Hinton et al., 2015), pruning (Liu et al., 2017; 2018), quantization (Zhou et al., 2016; Zhang et al., 2018), and sparsity (Wen et al., 2016). Quantization techniques have attracted particular attention because emerging hardware

[1] Hong Kong University of Science and Technology [2] Mohamed bin Zayed University of Artificial Intelligence [3] Jockey Club Institute for Advanced Study, HKUST [4] Reality Labs, Meta Inc [5] ACCESS - AI Chip Center for Emerging Smart Systems [6] Carnegie Mellon University. Correspondence to: Zhiqiang Shen <zhiqiangshen0214@gmail.com>.

*Proceedings of the 39th International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

accelerators (Judd et al., 2016; Jouppi et al., 2017; Sharma et al., 2018) begin to support low-precision inference.

To fully exploit the difference of representative capacity and redundancy in different parts (blocks, layers, and kernels) of deep neural networks, mixed precision quantization (MPQ) techniques have been put forward to assign different bitwidths to various components in a network. To achieve an MPQ network with satisfactory accuracy and compression, two critical problems must be solved consecutively: First, given a pre-trained network with weights $\{\boldsymbol{W}^{(l)}\}_{l=1}^{L}$ and a training dataset $\mathcal{D}$, how to find an optimal quantization strategy $\{\boldsymbol{b}^{(l)}\}_{l=1}^{L}$? Prior solutions addressed this general goal for MPQ are primarily based on searching or optimization. Whereas, the search-based methods which conduct an iterative scheme to explore the best structure cannot globally optimize the MPQ network for all modules and are also expensive to learn[1]. Meanwhile, existing optimization-based methods, such as FracBits (Yang & Jin, 2021), suffer from unstable optimization as the approximation design for gradients is not smooth and stable. Consequently, an ideal solution should be differentiable which enables automatic optimization of the bitwidth assignment in a global optimization space with more precise gradient approximations and smoother latent landscape as illustrated in Figure 1(d), which derives the primary motivation of this work.

The second problem stems from the quantization-aware post-training: after the quantization strategy $\{\boldsymbol{b}^{(l)}\}_{l=1}^{L}$ is acquired, how can we train a quantized network $\{\boldsymbol{W}_{\boldsymbol{b}^{(l)}}^{(l)}\}_{l=1}^{L}$ for maximizing the potential of performance and minimizing the accuracy sacrifice compared to a full-precision model? Empirically, during quantization training, the input information should be preserved and quantization error should be minimized through proper loss function design.

To address the above problems, in this work, we propose a novel **S**tochastic **D**ifferentiable **Q**uantization **(SDQ)** to tackle the challenges of searching and training MPQ. Concretely, we present an one-shot solution via representing the choice of discrete bitwidths as a set of Differentiable Bitwidth Parameters (DBPs), as shown in Fig. 1(a). DBPs are utilized in the forward path of the network as probability factors during the **stochastic quantization**. During the quantization strategy generation, DBPs will be updated to learn the optimal bitwidth assignment automatically. During the backpropagation, we use Gumbel-softmax reparameterization to estimate the gradient so the gradient can be back-propagated smoothly through our DBPs. Compared to previous differentiable solutions (Nikolić et al., 2020; Yang & Jin, 2021) using linear interpolation, the stochastic quantization scheme can substantially improve the network training stability and helps the DBPs converge with a smoother loss landscape shown in Fig. 1(d).

Addressing the challenge imposed by the second problem, we propose an entropy-aware bin regularizer based on entropy analysis, which regularizes the weights to keep more information-carrying components while considering the various precision of different layers. Knowledge distillation is also used to fully leverage the representation capability of full-precision teacher models.

We demonstrate the advantage of our SDQ in terms of effectiveness and efficiency on different networks and datasets. For ResNet18 on the ImageNet dataset, our quantized model can significantly improve top-1 accuracy (71.7%) compared to the full precision model (70.5%) with average bitwidth of both weights and activations no more than 4 bits. We also deploy ResNet18 on Bit Fusion (Sharma et al., 2018) accelerator to show the efficiency of our SDQ models.
In summary, our contribution can be concluded as:

- We present a novel SDQ framework to learn the optimal mixed precision quantization strategy via a set of differentiable bitwidth parameters as probability factors during the stochastic quantization.
- We utilize straight-through Gumbel-softmax estimator in the gradient computation *w.r.t.* differentiable bitwidth parameters. We also incorporate the quantization error regularization term while learning the mixed precision quantization strategy.
- We propose an entropy-aware bin regularizer to minimize the quantization error while considering mixed precision, which helps preserve more information-carrying components.
- We extensively evaluate our method on different network architectures and datasets. We further conduct **deployment experiments** on various hardware (GPUs and a real FPGA system) to demonstrate the effectiveness and efficiency of our models.

## 2. Related Work

Quantization techniques can be categorized into uniform and non-uniform quantization based on the quantization intervals. Non-uniform quantization (Miyashita et al., 2016; Zhang et al., 2018; Li et al., 2019b), due to its flexible representation, usually can better allocate the quantization values to minimize the quantization error and achieve better performance than uniform schemes. In addition, the quantization methods can also be classified as stochastic and deterministic quantization. Inspired by gradient estimation of stochastic neurons (Bengio et al., 2013), stochastic quantization (SQ) (Courbariaux et al., 2015; 2016; Dong et al., 2019a) has been explored. Unlike previous deterministic quantization, real-value variables of SQ are quantized to different quantization levels controlled by a probability distribution. For example, BinaryConnect (Courbariaux et al.,

[1] For $L$ layers with $B$ bitwidth candidate, searching both layer-wise weights and activations has a time complexity of $\mathcal{O}(B^{2L})$.

2015) transforms the real-value weights into +1 or -1 with probability determined by the distance to the zero point.

However, the aforementioned quantization schemes solely assign the same bitwidth to the entire model. Mixed precision quantization (MPQ), which employs different bitwidths in distinct layers or modules, can achieve a higher compression ratio and improved model capabilities. MPQ is a more promising direction in general, and it usually is divided into three categories: **Search-Based Methods**, **Metric-Based Methods**, and **Optimization-Based Methods**.

**Search-Based Methods** usually utilize Neural Architecture Search (Wu et al., 2018; Yu et al., 2020; Guo et al., 2020) or Reinforcement Learning (Wang et al., 2019; Elthakeb et al., 2019) to perform searching of quantization strategies. For instance, HAQ (Wang et al., 2019) leverages reinforcement learning and incorporates hardware feedback in the loop. DNAS (Wu et al., 2018) explore the quantized search space with gradient-based optimization to improve the efficiency. Despite these efforts to increase the efficiency of searching, the time and computational cost of generalizing search-based approaches to various network designs, datasets, and hardware platforms remain impediments.

**Metric-Based Methods** target at assigning bitwidth considering specific metrics that can be computed easily. HAWQ (Dong et al., 2019c;b; Yao et al., 2021) generates MPQ strategy based on the layers' Hessian spectrum. OMPQ (Ma et al., 2021) utilizes layer orthogonality to construct a linear programming problem to derive the bitwidth configuration. SAQ (Liu et al., 2021) determines the bitwidth configurations of each layer, encouraging lower bits for layers with lower sharpness. Although these methods are highly computation efficient, the generated MPQ strategies are usually sub-optimal as the mapping from these metrics to the bitwidths is manually established.

**Optimization-Based Methods** formulate the MPQ strategy generation problem as an optimization problem. The core challenge is to tackle the non-differentiability of task loss *w.r.t.* the bitwidth assignment. FracBits (Yang & Jin, 2021) proposed a fractional bitwidth parameter and used linear interpolation during the forward of quantization. DDQ (Zhaoyang et al., 2021) proposed a method to decompose the quantizer operation into the matrix-vector product. Uhlich et al. (Uhlich et al., 2020) proposed differentiable quantization via parametrization. These methods either introduce extra parameters such as quantization levels and the dynamic ranges of quantizers as the optimization target, or utilize linear interpolation between different quantization levels. The interpolation operation will introduce large number of useless optimization targets for quantization, and more instability is expected.

Noticeably, our work falls into the categories of **uniform**, **quantization-aware training**, and **optimization-based methods** of mixed precision network. To the best of our knowledge, SDQ is the first quantization strategy that adopts stochastic quantization to optimize the bitwidth assignment. Compared to previous research which also leverages the idea of differentiable bitwidth such as FracBits (Yang & Jin, 2021) and BitPruning (Nikolić et al., 2020), SDQ provides better training stability compared to the naive linear interpolation. Our DBPs can further denote discrete bitwidth candidates to better match the configuration of particular hardware accelerators, while others cannot, e.g., Bit Fusion (Sharma et al., 2018) only supports powers of 2 bitwidth.

# 3. Method

## 3.1. Preliminaries

In network quantization, real-valued weights $\boldsymbol{w}^r$ and activations $\boldsymbol{x}^r$ are converted to low-precision value $\boldsymbol{w}^q$ and $\boldsymbol{x}^q$. To mitigate the problem that gradient-based optimization cannot be directly applied to quantized value, previous research like DoReFa-Net (Zhou et al., 2016) exploits a straight-through estimator (STE) (Bengio et al., 2013) to assign the gradients passing in $\boldsymbol{x}^i$ and out $\boldsymbol{x}^o$ to be equal. Assume the loss function for the model is $\mathcal{L}$, an STE for $\boldsymbol{b}$-bit uniform quantizer $\boldsymbol{q}_b$ can be denoted as:

$$\begin{aligned} \textbf{Forward:}\ \ & \boldsymbol{x}^o = \boldsymbol{q}_b(\boldsymbol{x}^i) = \frac{1}{2^b-1}\text{round}[(2^b-1)\boldsymbol{x}^i]; \\ \textbf{Backward:}\ \ & \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^o} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^i}. \end{aligned} \tag{1}$$

Here $\boldsymbol{x}^i, \boldsymbol{x}^o \in [0,1]$. The weights and activations are first linearly transformed and clamped to interval $[0,1]$. The complete scheme for $b$-bit uniform quantizer $\boldsymbol{Q}_b$ is:

$$\boldsymbol{x}^q = \boldsymbol{Q}_b(\boldsymbol{x}^r) = \boldsymbol{q}_b\left(\frac{\tanh(\boldsymbol{x}^r)}{2\max(|\tanh(\boldsymbol{x}^r)|)} + \frac{1}{2}\right) - 1. \tag{2}$$

## 3.2. Generating Quantization Strategy

In the quantization strategy generation phase, our target is to learn the optimal bitwidth assignment. We introduce Differentiable Bitwidth Parameters (DBPs) $\boldsymbol{\beta}_{l,b_i}$ for each layer $\boldsymbol{l} \in \boldsymbol{L}$ and bitwidth candidates $\boldsymbol{b}_i \in \boldsymbol{B}_c$. DBPs are initialized to 1 to represent a deterministic quantization at different levels.

During forward, the weights can be quantized to two different bitwidths: the current bitwidth $\boldsymbol{b}_i$ and next bitwidth $\boldsymbol{b}_{i-1}$ in the candidate sets. The probability of quantization into two bitwidths is controlled by the DBP $\boldsymbol{\beta}_{l,b_i}$ at this layer.

$$\boldsymbol{w}^q_{l,b_i} = \begin{cases} \boldsymbol{Q}_{b_i}(\boldsymbol{w}^r_l) & \text{with probability } \ \boldsymbol{p} = \boldsymbol{\beta}_{l,b_i} \\ \boldsymbol{Q}_{b_{i-1}}(\boldsymbol{w}^r_l) & \text{with probability } \ \boldsymbol{p} = 1 - \boldsymbol{\beta}_{l,b_i} \end{cases}, \tag{3}$$

where $\boldsymbol{w}^q_{l,b_i}$ represents the quantized weight under $b_i$-bit stochastic-bitwidth quantizer. According to the characteristic of STE introduced in Eq. 1, the greadient through the quantization values and real values are the same: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^r_l} = \frac{\partial \mathcal{L}}{\partial Q_{b_i}(\boldsymbol{w}^r_l)} = \frac{\partial \mathcal{L}}{\partial Q_{b_{i-1}}(\boldsymbol{w}^r_l)}$. Thus, the expected gradients of quantized weight $\boldsymbol{w}^q_{l,b_i}$ over all trainable parameters in the network can be computed as:

$$\mathbb{E}[\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^q_{l,b_i}}] = \boldsymbol{\beta}_{l,b_i}\mathbb{E}[\frac{\partial \mathcal{L}}{\partial Q_{b_i}(\boldsymbol{w}^r_l)}] + (1-\boldsymbol{\beta}_{l,b_i})\mathbb{E}[\frac{\partial \mathcal{L}}{\partial Q_{b_{i-1}}(\boldsymbol{w}^r_l)}] = \mathbb{E}[\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^r_{l,b_i}}], \tag{4}$$

which is the same as the conventional STE. The derived gradient shows the proposed DBPs will not influence the gradient-based optimization of weight parameters during the training.

However, the real gradient still cannot backpropagate directly through the probability parameters $\boldsymbol{\beta}_{l,b_i}$. To tackle this problem, we use *Straight-Through Gumbel SoftMax Estimator* (Jang et al., 2016) as a reparameterization trick to allow gradients flow from quantized value $\boldsymbol{w}^q_{l,b_i}$ to DBPs $\boldsymbol{\beta}_{l,b_i}$. We can define the forwarding of previous stochastic-bitwidth quantization as $\boldsymbol{w}^q_{l,b_i} = \boldsymbol{c}^k_{l,b_i}\boldsymbol{Q}_{b_i}(\boldsymbol{w}^r_l) + (1-\boldsymbol{c}^k_{l,b_i})\boldsymbol{Q}_{b_{i-1}}(\boldsymbol{w}^r_l)$, where $\boldsymbol{c}^k_{l,b_i} \in \{0,1\}$ is a choice variable and it follows a Bernoulli distribution $\boldsymbol{c}^k_{l,b_i} \sim$ *Bernoulli*$(\boldsymbol{\beta}_{l,b_i})$. We use Gumbel SoftMax to transform the choice variable into a "soft" sampling operation while maintaining the distribution characteristics via:

$$\boldsymbol{c}^k_{l,b_i} = \frac{\exp((\log(\boldsymbol{\beta}_{l,b_i}) + \boldsymbol{g}^k)/\boldsymbol{\tau})}{\exp((\log(\boldsymbol{\beta}_{l,b_i}) + \boldsymbol{g}^{k_0})/\boldsymbol{\tau}) + \exp((\log(1-\boldsymbol{\beta}_{l,b_i}) + \boldsymbol{g}^{k_1})/\boldsymbol{\tau})}, \tag{5}$$

where $\boldsymbol{g}^k$, $\boldsymbol{g}^{k_0}$ and $\boldsymbol{g}^{k_1}$ are samples drawn from Gumbel(0,1) distribution, and $\boldsymbol{\tau}$ is the temperature coefficient to control the generated distribution. As has been proved in previous research (Jang et al., 2016), the gradient now can be easily computed and the expectation value of gradient *w.r.t.* quantized weights given in Eq. 4 still holds.

The advantage of MPQ is that different sensitivity of each layer to the quantization perturbation is fully considered. Previous research (Dong et al., 2019c) has shown that this sensitivity is closely aligned to the number of parameters and quantization error of this layer. In light of this, we propose a quantization-error regularizer to penalize layers with a large number of parameters and high quantization error:

$$\mathcal{L}_{QER} = \sum_{l_i}^{L} \boldsymbol{\beta}_{l_i,b_i}\lambda_{b_i}||\boldsymbol{w}^q_{l_i,b_i} - \boldsymbol{w}^r_{l_i}||^2_2, \tag{6}$$

where $\lambda_{b_i}$ is a weighting coefficient to balance between different bitwidth, and $||\boldsymbol{w}^q_{l_i,b_i} - \boldsymbol{w}^r_{l_i}||^2_2$ is the $L_2$ norm of quantization error. Noticeably, we do not normalize this $L_2$ norm term so it implies both the entries of weight parameters and the quantization error. It's also highly related to the bitwidth as it increase exponentially with the bitwidth. In our practice, we use $\lambda_{b_i} = (2^{|b_i|} - 1)^2$ to balance between different precision (please see Appendix. A for more details). We only optimize DBPs and will not optimize weight parameters with this quantization-error regularization. The complete optimization target during this stage can be denoted as:

$$\mathcal{O} = \mathop{\arg\min}_{\boldsymbol{W},\boldsymbol{\beta}} \mathcal{L}_{task} + \mathop{\arg\min}_{\boldsymbol{\beta}} \lambda_Q \mathcal{L}_{QER}, \tag{7}$$

where $\mathcal{L}_{task}$ is the task loss and $\lambda_Q$ is the coefficient to balance between task loss and quantization error regularization.

During the quantization strategy generation phase, we start from the highest precision in the bitwidth candidates sets, and progressively reduce the bitwidth. When DBP at layer $l$ satisfies the condition $\boldsymbol{\beta}_{l,b_i} < \boldsymbol{\beta}_{\mathrm{t}}$, the bitwidth assignment for layer $l$ will be reduced from $b_i$ to $b_{i-1}$ and we will start to optimize $\boldsymbol{\beta}_{l,b_{i-1}}$. After a pre-defined number of epochs for training, we can generate a MPQ strategy $\{b_{l_0}, \cdots b_{l_n}\}$ for each layer $l_i \in L$ based on the current DBPs $\{\boldsymbol{\beta}_{l_0,b_0}, \cdots \boldsymbol{\beta}_{l_n,b_n}\}$ that have been optimized.

### 3.3. Training with Quantization Strategy

While prior MPQ research (Uhlich et al., 2020; Zhaoyang et al., 2021) combined the searching and training stages, we choose a two-stage strategy since the optimization objectives are distinct in these two periods. In addition, more instability of training is expected when more quantization parameters are optimized simultaneously (*i.e.* DBPs) with the weight parameters. In light of this, we apply quantization-aware training with the generated MPQ strategy and without the DBPs. The total loss $\mathcal{L}$ for optimization in this stage is:

$$\mathcal{L} = \mathcal{L}_{KD} + \lambda_E \mathcal{L}_{EBR}, \tag{8}$$

where $\mathcal{L}_{KD}$ denotes the knowledge distillation loss and $\mathcal{L}_{EBR}$ denotes the entropy-preserving bin regularization. $\lambda_E$ is the weighting coefficient to balance between them. We will introduce them in Sec. 3.3.1 and Sec. 3.3.2, respectively.

#### 3.3.1. Knowledge Distillation

Intrinsically, a quantized classification network should learn an ideal similar mapping from input images to the output logits as a full-precision network, and the performance gap between them needs to be minimized. Based on this insight, we use knowledge distillation to train our MPQ network with a full-precision model as the teacher. The loss function is designed to enforce the similarity between the output distribution of full-precision teacher and MPQ student model:

$$\mathcal{L}_{KD} = -\frac{1}{n}\sum_{c}\sum_{i=1}^{n} p_c^{\mathcal{F}_\theta}(X_i)\log(p_c^{\mathcal{Q}_\theta}(X_i)) \tag{9}$$

where the KD loss is defined as the cross-entropy between the output distributions $p_c$ of a full-precision teacher $\mathcal{F}_\theta$

and a MPQ student $\mathcal{Q}_\theta$. $X_i$ is the input sample. $c$ and $n$ denote the classes and batch size, respectively. Note that this process can be regarded as the distribution calibration for the student network, and one-hot label is not involved.

#### 3.3.2. Entropy-aware Bin Regularization

In previous quantization methods, real-valued weights are usually distributed uniformly. As a result, the quantized weight might collapse to a few quantization bins close to zero. The information represented by the weights is heavily reduced during this process. In light of the information theory, entropy should be preserved in quantization to retain the representation capacity of neural networks.

In our method, after MPQ strategy is adopted, we propose **Entropy-aware Bin Regularization** to regularize weights in group of different bitwidth. The entropy carried by quantized weight with a specific bit width $b$ can be denoted as $\mathcal{H}_b(\boldsymbol{W}) = -\boldsymbol{p}_b(\boldsymbol{w}_i)\log(\boldsymbol{p}_b(\boldsymbol{w}_i)), s.t. \sum_{i=1}^{2^b} \boldsymbol{p}_b(\boldsymbol{w}_i) = 1$, where $\boldsymbol{p}_b(\boldsymbol{w}_i)$ indicates the proportion of weights quantized to $i$-th quantization bin. We can derive that the entropy $\mathcal{H}_b(\boldsymbol{W})$ is maximized when $\boldsymbol{p}_b^*(\boldsymbol{w}_i) = 1/2^b$ for all $i \in \{1, \ldots 2^b\}$. In light of the entropy analysis and inspired by previous research (Li et al., 2019b; Yamamoto, 2021), we first normalize real-valued weights $\boldsymbol{w}_l^r$ of layer $l$ with bitwidth $b$ to $\boldsymbol{w}_l^{r*} = \frac{2^{(b-1)}}{2^b-1} \frac{|\boldsymbol{w}_l^r|}{||\boldsymbol{w}_l^r||_1} \boldsymbol{w}_l^r$. The corresponding quantized weights $\boldsymbol{w}_l^q$ are approximately uniformly distributed in all quantization levels. Here, $|\boldsymbol{w}_l^r|$ is the number of entries in $\boldsymbol{w}_l^r$ and $||\boldsymbol{w}_l^r||_1$ computes the $L_1$-norm of $\boldsymbol{w}_l^r$.

Different from weight normalization in APOT (Li et al., 2019b) and LCQ (Yamamoto, 2021) which only consider the global weight distribution, we also want to minimize the quantization error in a specific quantization bin to reduce the information loss. Therefore, we regularize the distribution of the weights in a quantization bin to be a "sharp" Gaussian distribution (i.e., a Dirac delta distribution ideally): its mean approaching quantization value and variance approaching zero. Motivated by these considerations, the Entropy-aware Bin Regularization is formulated as:

$$\mathcal{L}_{EBR} = \sum_{l=1}^{L} \sum_{n=1}^{2^{b(l)}} \mathcal{L}_{mse}(\overline{\boldsymbol{w}_{n,l}^r}, \boldsymbol{w}_{n,l}^q) + \mathcal{V}(\boldsymbol{w}_{n,l}^r), \quad (10)$$

where $\boldsymbol{w}_{n,l}^r, \boldsymbol{w}_{n,l}^q$ represents the real value and quantized value of weights in layer $l$ and quantization bin $n$. $\mathcal{L}_{mse}$ computes the mean square error, $\overline{(\cdot)}$ computes the mean, and $\mathcal{V}(\cdot)$ computes variance for all quantization bins with more than two elements.

### 3.4. Method Analysis and Discussion

The complete algorithm of our proposed SDQ is summarized in Alg. 1. Since our SDQ only performs stochastic quantization on weights, the precision of activations is unaffected throughout the quantization strategy generation phase. We freeze the activation bitwidth with a fixed value for the entire network during the training phase, and the activation is quantized in the same way as denoted by Eq. 2. Also note that our stochastic quantization is only performed on quantization strategy generation phase and is not applied during post-training or real inference.

While we only demonstrate how to use SDQ on layer-wise quantization. Our approach can easily be adapted to quantize models at many levels of granularity, such as block-wise, net-wise, and kernel-wise. However, the performance will be inhibited when using SDQ on coarse-grained granularity, such as blocks and networks, since the sensitivity difference between each layer will not be fully exploited. Additional training instability is expected when SDQ is used to fine-grained kernel-wise quantization because more quantization parameters are optimized alongside weight parameters during the MPQ strategy learning phase. Furthermore, current hardware accelerators cannot implement kernel-wise quantization, limiting the actual performance when deploying quantized models. Appendix. B contains further studies demonstrating that SDQ for layer-wise quantization produces the best results.

**Algorithm 1** Stochastic Differentiable Quantization

**Input:** the network with full precision weight $\{\boldsymbol{W}^r\}_{l=1}^L$, differentiable bitwidth parameters $\{\boldsymbol{\beta}\}_{l=1}^L$, bitwidth candidate sets $\mathcal{B}$, maximum training epoch $\boldsymbol{E}_{\text{G}}$ (epoch for generating quantization strategy) and $\boldsymbol{E}_{\text{T}}$ (epoch for post-training), threshold $\boldsymbol{\beta}_{\text{t}}$ for decaying bitwidth

**Output:** quantized network with weight $\{\boldsymbol{W}^q\}_{l=1}^L$ and bitwidth allocation $\{\boldsymbol{b}^{(l)}\}_{l=1}^L$

```
// Phase 1: Generating Quantization Strategy
for $l = 1$ to $L$, $\boldsymbol{b} \in \mathcal{B}$ do
    Initialize differentiable bitwidth parameters $\boldsymbol{\beta}_{l,b} = 1$
end for
for epoch = 1 to $\boldsymbol{E}_{\text{G}}$ do
    Quantize real-value weight $\boldsymbol{W}^r$ to $\boldsymbol{W}^q$ by Eq. 3
    Compute the task loss $\mathcal{L}_{task}$ and $\mathcal{L}_{QER}$ by Eq. 6
    Compute gradient $\nabla \boldsymbol{W}^r$ and $\nabla \boldsymbol{\beta}$, update parameters
    Update $\{\boldsymbol{b}^{(l)}\}_{l=1}^L$ according to $\{\boldsymbol{\beta}\}_{l=1}^L$ and $\boldsymbol{\beta}_{\text{t}}$
end for
Generate MPQ strategy $\{\boldsymbol{b}^{(l)}\}_{l=1}^L$ from DBPs $\{\boldsymbol{\beta}\}_{l=1}^L$
// Phase 2: Post-training with Quantization Strategy
for epoch = 1 to $\boldsymbol{E}_{\text{T}}$ do
    Quantize real-value weights and activations by Eq. 2
    Compute the knowledge distillation loss $\mathcal{L}_{KD}$ and
    regularization $\mathcal{L}_{EBR}$ by Eq. 9 and Eq. 10
    Compute gradient $\nabla \boldsymbol{W}^r$, update parameters
end for
Return MPQ network $\{\boldsymbol{W}^q\}_{l=1}^L$ with strategy $\{\boldsymbol{b}^{(l)}\}_{l=1}^L$
```

# 4. Experiments

To evaluate the effectiveness of the proposed SDQ, we conduct experiments on the CIFAR and ImageNet datasets. We first introduce the dataset, network, and training strategy in Sec. 4.1, followed by the comparison to state-of-the-art quantization methods in Sec. 4.2. We then analyze the effect and role of each proposed component of SDQ in Sec. 4.3. In Sec. 4.4, we visualize the MPQ strategy and how our proposed SDQ improves the representation capacity of the quantized model. Hardware deployment experiments on different devices are designed to demonstrate the energy and time efficiency of our models in Sec. 4.5.

## 4.1. Experimental Settings

**Dataset** The experiments are carried out on CIFAR-10 dataset (Krizhevsky et al., 2009) and ImageNet dataset (Deng et al., 2009). We only perform basic data augmentation in PyTorch (Paszke et al., 2019), which includes *RandomResizedCrop* and *RandomHorizontalFlip* during training, and single-crop operation during evaluation.

**Network** We evaluate ResNet20 on CIFAR-10, and evaluate ResNet18 and MobileNetV2 on ImageNet dataset. Due to the fact that the first and last layers are more sensitive to quantization perturbation compared to intermediate layers, we fix the bitwidth of them following previous work (Yang & Jin, 2021).

**Training detail** Following previous quantization methods (Zhou et al., 2016; Jung et al., 2019), we adopt real-value pre-trained weights as initialization. We train and evaluate our models on various hardware platforms, including various NVIDIA GPUs and FPGA. Details of all hyper-parameters and training schemes are shown in Appendix. C.

## 4.2. Comparison with State-of-the-Art Method

Table. 1 compare our SDQ with existing methods for ResNet20 on CIFAR-10. We can see that our SDQ models significantly outperform the previous fixed precision quantization methods while also yielding better accuracy than other MPQ models with fewer average bitwidth.

*Table 1.* Comparison with State-of-the-Art mixed precision quantization methods (ResNet20 on CIFAR-10).

| Method | Bit-width | mixed | Accuracy(%) | | WCR |
|---|---|---|---|---|---|
| | (W/A) | | Top-1 | FP Top-1 | |
| Dorefa(Zhou et al., 2016) | 2/32 | | 88.2 | 92.4 | 16× |
| PACT (Choi et al., 2018) | 2/32 | | 89.7 | 92.4 | 16× |
| LQ-net(Zhang et al., 2018) | 2/32 | | 91.1 | 92.4 | 16× |
| TTQ (Jain et al., 2019) | 2.00/32 | ✓ | 91.2 | 92.4 | 16× |
| Uhlich et al. (Uhlich et al., 2020) | 2.00/32 | ✓ | 91.4 | 92.4 | 16× |
| BSQ (Yang et al., 2020) | 2.08/32 | ✓ | 91.9 | 92.6 | 15.4× |
| DDQ (Zhaoyang et al., 2021) | 2.00/32 | ✓ | 91.6 | 92.4 | 16× |
| Ours | 1.93/32 | ✓ | **92.1** | 92.4 | **16.6**× |

Table. 2 shows the ImageNet classification performance of our SDQ on ResNet18 and MobileNetV2. Since different methods adopt different full-precision (FP) models as initialization, we also report the corresponding Top-1 accuracy for the FP model for different methods.

Compared to the baseline FP model, our SDQ-trained models perform comparably or even better when quantized to low precision. For example, our ResNet18 achieves 71.7% Top-1 accuracy when the average bitwdith for weights and activations is 3.61 and 4 respectively, which has a 1.2% absolute gain on the full-precision model. Our MobileNetV2 model is **the first quantized models** with accuracy higher than the FP initialization (72.0% vs. 71.9%), and average precision for weights and activations are lower than 4.

With an optimal MPQ strategy and an effective training scheme, our SDQ outperforms previous quantization methods significantly under the same bitwidth configuration. To the best of our knowledge, the highest accuracy reported by previous uniform methods with weights and activation bitwidths not more than 4 bits are 70.6% on ResNet18 and 71.6% on MobileNetV2 from FracBits-SAT (Yang & Jin, 2021). Our SDQ achieve an increase of 1.1% on ResNet18 and 0.3% on MobileNetV2 with a more compact bitwidth (3.61/4 vs. 4/4 on ResNet18, 3.66/4 vs. 4/4 on MobileNetV2). Note that SDQ even outperforms state-of-the-art non-uniform quantization methods (Li et al., 2019b; Chang et al., 2021; Zhaoyang et al., 2021), proving the effectiveness of our methods.

## 4.3. Ablation Study

We further conduct ablation studies to prove the contribution of different components to our model performance. These components include SDQ quantization strategy generation, knowledge distillation, and entropy-aware bin regularization. Table. 3 compares different quantization strategy generation schemes fairly on the same initialization and training. The performance of the quantized model with the quantization strategy generated by our SDQ is very close to previous MPQ models, while the average bitwidths of our strategy are lower. This comparison proves that our strategy generation contributes to the improvement of accuracy.

*Table 3.* Comparison with State-of-the-Art mixed precision quantization strategy generation methods under same training and initialization (MobileNetV2 on ImageNet).

| MPQ Strategy | Bit-width | Accuracy(%) | |
|---|---|---|---|
| | (W/A) | Top-1 | Top-5 |
| Uhlich et al. (Uhlich et al., 2020) | 3.75/4 | 71.8 | 90.3 |
| FracBits (Yang & Jin, 2021) | 4/4 | 72.0 | 90.4 |
| Our strategy | **3.66/4** | **72.0** | **90.5** |

Comparison of different weight regularization methods are shown in Table. 4. The experiments are conducted on MPQ ResNet18 with an average bitwidth of 3.61 and 2 for weights and activations. The insights of our proposed EBR are to penalize the global distribution of weights to be too centralized

*Table 2.* Comparison with State-of-the-Art mixed precision quantization methods (ResNet18 and MobileNetV2 on ImageNet). Bitwidth (W/A) denotes the average bitwidth for weights and activation parameters. WCR represents the weight compression rate. BitOPs denotes the bit operations. For a filter $f$, the BitOPs is defined as BitOPs$(f) = b_w b_a |f| w_f h_f / s_f^2$, where $b_w$ and $b_a$ are the bitwidths for weights and activations, $|\cdot|$ denotes the cardinality of the filter, $w_f$,$h_f$,$s_f$ are the spatial width, height, and stride of the filter.

| Network | Method | Bit-width (W/A) | Mixed | Uniform | Accuracy (%) Top-1 | Accuracy (%) FP Top-1 | WCR | Model Size (MB) | BitOPs(G) |
|---|---|---|---|---|---|---|---|---|---|
| ResNet18 | Dorefa†(Zhou et al., 2016) | 4/4 | | ✓ | 68.1 | 70.5 | 8× | 5.8 | 35.2 |
| | PACT†(Choi et al., 2018) | 4/4 | | ✓ | 69.2 | 70.5 | 8× | 5.8 | 35.2 |
| | LQ-net (Zhang et al., 2018) | 4/4 | | | 69.3 | 70.5 | 8× | 5.8 | 35.2 |
| | APOT (Li et al., 2019b) | 4/4 | | | 70.7 | 70.5 | 8× | 5.8 | 34.7 |
| | DNAS†(Wu et al., 2018) | -/- | ✓ | ✓ | 70.6 | 71.0 | 8× | 5.8 | 35.2 |
| | HAQ (Wang et al., 2019) | 4/32 | ✓ | ✓ | 70.4 | 70.5 | 8× | 5.8 | 465 |
| | EdMIPS (Cai & Vasconcelos, 2020) | 4/4 | ✓ | ✓ | 68.0 | 70.2 | 8× | 5.8 | 34.7 |
| | HAWQ-V3†(Yao et al., 2021) | 4.8/7.5 | ✓ | ✓ | 70.4 | 71.5 | 6.7× | 7.0 | 72.0 |
| | Chen et al. (Chen et al., 2021) | 3.85/4 | ✓ | ✓ | $69.7_{\downarrow 0.1\%}$ | 69.8 | 8.3× | 5.6 | 33.4 |
| | FracBits-SAT (Yang & Jin, 2021) | 4/4 | ✓ | ✓ | $70.6_{\uparrow 0.4\%}$ | 70.2 | 8× | 5.8 | 34.7 |
| | Uhlich et al. (Uhlich et al., 2020) | 3.88/4 | ✓ | | 70.1 | 70.3 | 8.3× | 5.6 | 33.7 |
| | RMSMP (Chang et al., 2021) | 4/4 | ✓ | | 70.7 | 70.3 | 8× | 5.8 | 34.7 |
| | DDQ (Zhaoyang et al., 2021) | 4/4 | ✓ | | 71.2 | 70.5 | 8× | 5.8 | 34.7 |
| | **Ours** | 3.61/8 | ✓ | ✓ | $\mathbf{72.1}_{\uparrow \mathbf{1.6\%}}$ | 70.5 | **8.9**× | 5.2 | 62.6 |
| | | 3.61/4 | ✓ | ✓ | $\mathbf{71.7}_{\uparrow \mathbf{1.2\%}}$ | 70.5 | **8.9**× | 5.2 | **31.3** |
| | | 3.61/3 | ✓ | ✓ | $\mathbf{70.2}_{\downarrow \mathbf{0.3\%}}$ | 70.5 | **8.9**× | 5.2 | **23.5** |
| | | 3.61/2 | ✓ | ✓ | $\mathbf{69.1}_{\downarrow \mathbf{1.4\%}}$ | 70.5 | **8.9**× | 5.2 | **15.7** |
| MobileNetV2 | Dorefa†(Zhou et al., 2016) | 4/4 | | ✓ | 61.8 | 71.9 | 8× | 1.8 | 7.42 |
| | PACT†(Choi et al., 2018) | 4/4 | | ✓ | 61.4 | 71.9 | 8× | 1.8 | 7.42 |
| | LQ-net (Zhang et al., 2018) | 4/4 | | | 64.4 | 71.9 | 8× | 1.8 | 7.42 |
| | APOT (Li et al., 2019b) | 4/4 | | | 71.0 | 71.9 | 8× | 1.8 | 5.35 |
| | HAQ (Wang et al., 2019) | 4/32 | ✓ | ✓ | 71.5 | 71.9 | 8× | 1.8 | 42.8 |
| | HMQ (Habi et al., 2020) | 3.98/4 | ✓ | ✓ | 70.9 | 71.9 | 8.1× | 1.7 | 5.32 |
| | Chen et al. (Chen et al., 2021) | 4.27/8 | ✓ | ✓ | $71.8_{\downarrow 0.1\%}$ | 71.9 | 7.5× | 1.9 | 5.32 |
| | FracBits-SAT (Yang & Jin, 2021) | 4/4 | ✓ | ✓ | $71.6_{\downarrow 0.2\%}$ | 71.8 | 8× | 1.8 | 5.35 |
| | Uhlich et al. (Uhlich et al., 2020) | 3.75/4 | ✓ | | 69.8 | 70.2 | 8.5× | 1.6 | 5.01 |
| | RMSMP (Chang et al., 2021) | 4/4 | ✓ | | 69.0 | 71.9 | 8× | 1.8 | 5.35 |
| | DDQ (Zhaoyang et al., 2021) | 4/4 | ✓ | | 71.8 | 71.9 | 8× | 1.8 | 5.35 |
| | **Ours** | 3.66/8 | ✓ | ✓ | $\mathbf{72.9}_{\uparrow \mathbf{1.0\%}}$ | 71.9 | **8.7**× | 1.8 | 9.79 |
| | | 3.66/4 | ✓ | ✓ | $\mathbf{72.0}_{\uparrow \mathbf{0.1\%}}$ | 71.9 | **8.7**× | 1.8 | **4.89** |

† re-implementation for fair comparisons under the same backbone architectures

and penalize the local distribution of weights in different quantization bins to be too dispersed. The effect is also shown in Fig. 5. Compared to other weights regularization methods which only consider the global distribution, our EBR significantly improves the quantization robustness of MPQ models.

*Table 4.* Comparison of our Entropy-aware Bin Regularization (EBR) and other weight regularization methods for our mixed precision quantized ResNet18 on ImageNet.

| Method | | Top-1 Acc | Top-5 Acc |
|---|---|---|---|
| Baseline | | 67.6 | 87.6 |
| Weight Norm (Salimans & Kingma, 2016) | | 66.6 | 86.7 |
| KURE (Shkolnik et al., 2020) | | 68.5 | 88.4 |
| EBR | $\lambda_E = 0.01$ | 68.6 | 88.4 |
| | $\lambda_E = 0.1$ | **69.1** | **88.5** |
| | $\lambda_E = 1$ | 68.9 | 88.4 |

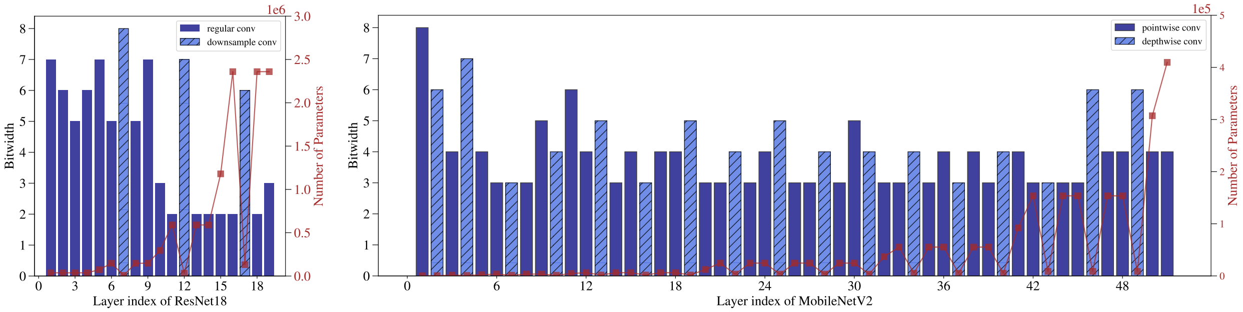
How the knowledge distillation contributes to the training of MPQ models are shown in Table. 5. Learning from FP models helps minimize the performance gap of quantization models. When the performance of FP teacher models is better, the performance of student models also improves.

*Table 5.* Comparison of different teacher models of knowledge distillation for our mixed precision quantized ResNet18 on ImageNet. One-hot label with CE loss is used in the experiment w/o KD.

| Method | Teacher | Top-1 Acc | Top-5 Acc |
|---|---|---|---|
| Ours w/o KD | Ground Truth | 70.5 | 89.5 |
| Ours | ResNet34 | 70.7 | 89.7 |
| | ResNet50 | 71.1 | 89.9 |
| | ResNet101 | **71.7** | **90.2** |

### 4.4. Visualization and Analysis

To intuitively demonstrate the bitwidth assignment generated by our SDQ, the quantization strategy of weights in different layers of ResNet18 and MobileNetV2 is visualized in Fig. 2(a) and Fig. 2(b) respectively. Our SDQ learns more bits for those layers with fewer parameters (*downsample conv* layers in ResNet18 and *depthwise conv* layers in MobileNetV2), which proves that more redundancy is expected for those layers. As layers with the same number of parameters can have different optimal bitwidth, it can be proved that there is no simple heuristic to allocate bitwidth. We can also see from the bitwidth assignment that the first few layers and the last few layers has higher bitwidth. How



(a) ResNet18 (3.61bit) (b) MobileNetV2 (3.66bit, The first 3×3 conv layer and last fc layer is not visualized here)

*Figure 2.* Mixed Precision Quantization strategy generated by our SDQ for ResNet18 and MobileNetV2 on ImageNet dataset.

bitwidth assignment evolves during the quantization strategy generation phase is shown in Fig. 3. While the decay of bitwidth is slow for *downsample conv* layers, the regular *conv* layers decay quickly at high precision and slow down when it reaches low precision.

Compared to uniform quantization, the MPQ network performs better with fewer bits because the MPQ network fully utilizes the difference of layers' sensitivity to quantization. Fig. 4(a) and Fig. 4(b) visualizes the feature embedding of last conv layer in ResNet20. As expected, the clusters of dog and deer are mixed up together in the baseline model with 2-bit uniform quantization. While the model of SDQ can split those classes sharing similar representations.

We proposed entropy-aware bin regularization to preserve the entropy and minimize the quantization error. Fig. 5 shows the histogram of full-precision weight distribution and the distribution after quantization to 2-bit. We can observe that the weights are more centralized to the quantization point in our models compared to baseline. Further, the number of weights quantized to each level in our quantized models is closer compared to baseline.

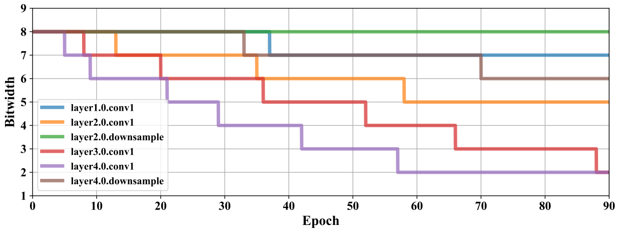
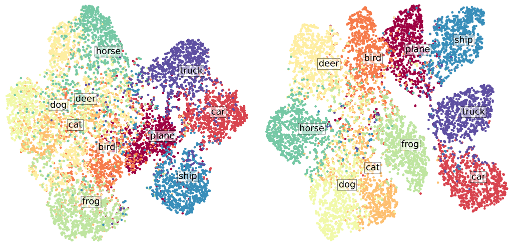


*Figure 3.* Bitwdith assignment of selected layers in ResNet18 during quantization strategy generation.

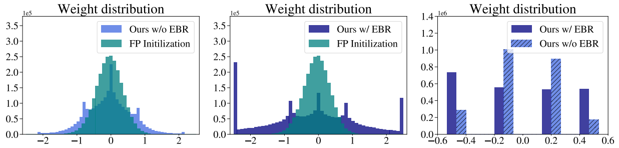
### 4.5. Hardware Efficiency on Accelerator

We further conduct hardware experiments to evaluate the efficiency of our MPQ model. In Table 6, we evaluate our model on accelerator supporting mixed precision arithmetic operation: Bit Fusion (Sharma et al., 2018). Bit Fusion only supports multiplications of power-of-two bits (*i.e.* 2,4,8,16 bits), so there is a performance gap between theoretical compression and real compression. Our ResNet18 model with an average weight bitwidth of 3.61 achieves much better accuracy than the ResNet18 model with a weight bitwidth of 4 and surpasses its time and energy efficiency.



(a) Baseline (2 bit) (b) Mixed Ours (1.93 bit)

*Figure 4.* Comparison of feature embedding visualization using t-SNE on CIFAR-10 evaluation set. The feature embedding are extracted from the output of last conv layer of ResNet20. The performance of these two models are reported in Table. 1.



(a) Weights w/o EBR (b) Weights w/ EBR (c) Quantization

*Figure 5.* Histogram of the weight distribution and quantization distribution. 5(a) and 5(b) are the histogram of weights of *layer4.1.conv2* with 2-bit precision of our quantized ResNet18 model with and without EBR respectively. 5(c) compares the quantization results with and without EBR.

*Table 6.* Comparison of latency and energy consumption of quantized ResNet18 model on Bit Fusion.

| Method | Bitwidth(W/A) | Top-1 Acc | Latency | Energy |
|---|---|---|---|---|
| Dorefa | 4/8 | 69.8 | 48.99 ms | 93.34 mJ |
| Ours | 3.61/8 | **72.1** | 46.18 ms | 90.18 mJ |
| Dorefa | 4/4 | 67.1 | 34.61 ms | 64.16 mJ |
| Ours | 3.61/4 | **71.1** | 32.84 ms | 60.49 mJ |
| Dorefa | 4/2 | 63.6 | 30.77 ms | 54.08 mJ |
| Ours | 3.61/2 | **69.1** | 28.18 ms | 49.05 mJ |

*Table 7.* Performance comparison on COCO detection benchmark.

| Method | Bitwidth (W/A) | Hardware | AP | AP$_{50}$ | AP$_{75}$ | AP$_S$ | AP$_M$ | AP$_L$ | Latency | Energy | FPS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLOv4-tiny (Bochkovskiy et al., 2020) | 32/32 | GPU | 20.4 | 38.7 | 19.5 | 7.4 | 24.1 | 27.1 | - | - | - |
| Dorefa (Zhou et al., 2016) | 8/8 | FPGA | 16.1 | 32.3 | 14.5 | 4.3 | 17.3 | 24.7 | 34.18ms | 268.3mJ | 29 |
| | 4/4 | FPGA | 15.4 | 29.2 | 12.5 | 3.7 | 16.9 | 23.3 | 18.64ms | 146.3mJ | 53 |
| Ours | 3.88/4 | FPGA | 15.9 | 31.1 | 14.7 | 4.2 | 17.6 | 24.2 | 21.28ms | 167.1mJ | 47 |

### 4.6. COCO Detection Hardware Experiment

We also evaluate our SDQ on object detection tasks which are more challenging. We implement the mixed-precision compact YOLOv4-tiny (Bochkovskiy et al., 2020) with our SDQ on COCO detection dataset (Lin et al., 2014), which consists of 80 object categories, 330K images, and 1.5M instances. Different from previous research (Li et al., 2019a; Wang et al., 2020) applying quantization on Faster R-CNN (Ren et al., 2015) or RetinaNet (Lin et al., 2017) using ResNet and MobileNet as backbones, YOLOv4-tiny is already very compact and more sensitive to quantization. The quantization strategy generation and training phase are conducted on COCO-2017-TRAIN partition. During the training phase, percentile activation calibration (Li et al., 2019a) is used to discard outlier activation values. Since our target deployment platform only supports power-of-two bitwidth, we set the bitwidth candidate $\mathcal{B} = \{1, 2, 4, 8\}$. The input size for the object detector is $416 \times 416$. Following the standard COCO evaluation metric, we report the average precision (AP) for different IoU thresholds on the COCO-2017-VAL partition. The results, including latency and energy consumption deploying on an FPGA system, are shown in Table. 7.

From the results we can see that our SDQ improves the efficiency of object detectors without incurring significant mAP loss compared to baseline quantization methods. While our object detector has bitwidths not more than 4 for both weights and activations, it outperforms the 4-bit quantized model and can even compete with the 8-bit quantization of the object detector. For the FPGA deployment statistics, our SDQ demonstrates good hardware affinity yielding latency and energy consumption close to 4-bit uniform single-precision models. Overall, mixed-precision YOLOv4-tiny trained with our SDQ can achieve the performance of 8-bit quantized models while retaining the hardware efficiency of 4-bit quantized models.

#### 4.6.1. FPGA System Setting

The system is implemented on the Xilinx U50 FPGA platform and consumes 259688 LUTs and 210.5 BRAM. As shown in Fig. 6, a CNN accelerator system is comprised of multiple cores, controller, on-chip memory, downloader, and uploader. Each core consists of 4×16 dedicated array of INT-8 multiply-and-accumulate (MAC) processing elements. All cores share the same on-chip memory, which stores the input feature map (ifmap), weight, index, and output feature map (ofmap). The index is used for weight sparse to scratch the input activation. The controller module uses instructions to take charge of convolution computation, data download, and upload.

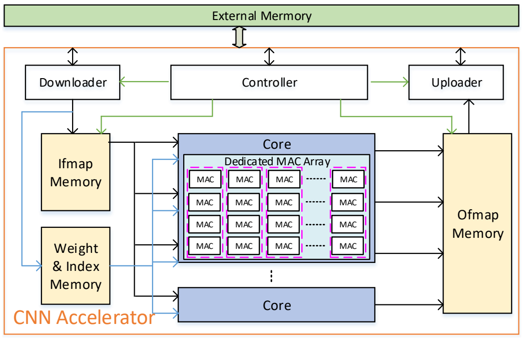


*Figure 6.* System architecture schematic. The parameters of our hardware experiment platform are: MAC array-4 rows × 16 columns, Frequency-200MHz, Number of Cores-8.

## 5. Conclusion

We have presented **S**tochastic **D**ifferentiable **Q**uantization **(SDQ)**, a learning based mixed precision quantization framework that can optimize the optimal MPQ strategy automatically. In contrast to previous MPQ methods, SDQ introduces a set of Differentiable Bitwidth Parameters (DBPs) that follows a stochastic quantization scheme to make DBPs differentiable with a stabler and smoother optimization process. Since SDQ is optimized on a global searching space, the learned MPQ strategy is usually better than other competitors. Moreover, Quantization Error Regularization (QER) and Entropy-aware Bin Regularization (EBR) are integrated into the MPQ learning and post-training stages to minimize the quantization error. We demonstrate the superiority of our SDQ with comprehensive experiments on different hardware platforms such as GPUs and FPGA across various networks. SDQ achieves state-of-the-art accuracy over previous quantization methods with fewer average bits. Extensive hardware experiments prove the superior efficiency of our models deployed on mixed-precision accelerators.

## Acknowledgements

This research was partially supported by ACCESS - AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

## References

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Cai, Z. and Vasconcelos, N. Rethinking differentiable search for mixed-precision neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2349–2358, 2020.

Chang, S.-E., Li, Y., Sun, M., Jiang, W., Liu, S., Wang, Y., and Lin, X. Rmsmp: A novel deep neural network quantization framework with row-wise mixed schemes and multiple precisions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5251–5260, 2021.

Chen, W., Wang, P., and Cheng, J. Towards mixed-precision quantization of neural networks via constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5350–5359, 2021.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I.-J., Srinivasan, V., and Gopalakrishnan, K. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 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.

Dong, Y., Ni, R., Li, J., Chen, Y., Su, H., and Zhu, J. Stochastic quantization for learning accurate low-bit deep neural networks. *International Journal of Computer Vision*, 127 (11):1629–1642, 2019a.

Dong, Z., Yao, Z., Cai, Y., Arfeen, D., Gholami, A., Mahoney, M. W., and Keutzer, K. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *arXiv preprint arXiv:1911.03852*, 2019b.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019c.

Elthakeb, A., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., Gao, S., and Esmaeilzadeh, H. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. In *NeurIPS ML for Systems workshop, 2018*, 2019.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Habi, H. V., Jennings, R. H., and Netzer, A. Hmq: Hardware friendly mixed precision quantization block for cnns. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pp. 448–463. Springer, 2020.

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.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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. *arXiv preprint arXiv:1704.04861*, 2017.

Jain, S. R., Gural, A., Wu, M., and Dick, C. H. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *arXiv preprint arXiv:1903.08066*, 2019.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

Judd, P., Albericio, J., Hetherington, T., Aamodt, T. M., and Moshovos, A. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12. IEEE, 2016.

Jung, S., Son, C., Lee, S., Son, J., Han, J.-J., Kwak, Y., Hwang, S. J., and Choi, C. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6391–6401, 2018.

Li, R., Wang, Y., Liang, F., Qin, H., Yan, J., and Fan, R. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2810–2819, 2019a.

Li, Y., Dong, X., and Wang, W. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019b.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Liu, J., Cai, J., and Zhuang, B. Sharpness-aware quantization for deep neural networks. *arXiv preprint arXiv:2111.12273*, 2021.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2018.

Ma, Y., Jin, T., Zheng, X., Wang, Y., Li, H., Jiang, G., Zhang, W., and Ji, R. Ompq: Orthogonal mixed precision quantization. *arXiv preprint arXiv:2109.07865*, 2021.

Miyashita, D., Lee, E. H., and Murmann, B. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

Nikolić, M., Hacene, G. B., Bannon, C., Lascorz, A. D., Courbariaux, M., Bengio, Y., Gripon, V., and Moshovos, A. Bitpruning: Learning bitlengths for aggressive and accurate quantization. *arXiv preprint arXiv:2002.03090*, 2020.

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. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104. PMLR, 2018.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909, 2016.

Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Chandra, V., and Esmaeilzadeh, H. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 764–775. IEEE, 2018.

Shkolnik, M., Chmiel, B., Banner, R., Shomron, G., Nahshan, Y., Bronstein, A., and Weiser, U. Robust quantization: One model to rule them all. *arXiv preprint arXiv:2002.07686*, 2020.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J. A., Tiedemann, S., Kemp, T., and Nakamura, A. Mixed precision dnns: All you need is a good parametrization. In *International Conference on Learning Representations*, 2020.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Wang, Z., Wu, Z., Lu, J., and Zhou, J. Bidet: An efficient binarized object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2049–2058, 2020.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29:2074–2082, 2016.

Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., and Keutzer, K. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

Yamamoto, K. Learnable companding quantization for accurate low-bit neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5029–5038, 2021.

Yang, H., Duan, L., Chen, Y., and Li, H. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. In *International Conference on Learning Representations*, 2020.

Yang, L. and Jin, Q. Fracbits: Mixed precision quantization via fractional bit-widths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10612–10620, 2021.

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 *International Conference on Machine Learning*, pp. 11875–11886. PMLR, 2021.

You, Y. *Audio Coding: Theory and Applications*. Springer Science & Business Media, 2010.

Yu, H., Han, Q., Li, J., Shi, J., Cheng, G., and Fan, B. Search what you want: Barrier panelty nas for mixed precision quantization. In *European Conference on Computer Vision*, pp. 1–16. Springer, 2020.

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.

Zhaoyang, Z., Wenqi, S., Jinwei, G., Xiaogang, W., and Ping, L. Differentiable dynamic quantization with mixed precision and adaptive resolution. *arXiv preprint arXiv:2106.02295*, 2021.

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

# Appendix

This appendix includes additional analysis, implementation details, and extended experimental analysis not included in the main text due to space limitation. These contents are organized in separate sections as follows:

- Sec. A analyzes how quantization error varies within different layers with mixed precision and how we propose a weighting coefficient to balance between these precision.
- Sec. B elaborates the performance of MPQ in different granularities of networks and the reasons why layer-wise quantization is optimal.
- Sec. C includes the details of training and experiments, which includes the hyper-parameters, the training dynamics, and the detailed compression of bitwidth assignment shown in Table.

## A. Quantization Error Analysis

Quantization error $\boldsymbol{\Omega}$ is defined as L2 distance $||\boldsymbol{x}^q - \boldsymbol{x}^r||_2$ between quantized value $\boldsymbol{x}^q$ and real value $\boldsymbol{x}^r$. In uniform quantization, $\boldsymbol{\Omega}_u$ depends on the characteristics of layers which includes parameter amounts, learned weight distribution, and bitwidth assignment. Among these factors, the most important one is the bitwidth of the layers. As shown in Table. 8, the squared quantization error $\boldsymbol{\Omega}_u^2$ grows exponentially with the bitwidth.

*Table 8.* Comparison of squared quantization error $\boldsymbol{\Omega}_u^2$ for various bitwidth of different layers in ResNet20

| Layer | Parameter Size | 8-bit | 6-bit | 4-bit | 3-bit | 2-bit |
|---|---|---|---|---|---|---|
| Layer1.2.conv1 | 2.3e4 | 0.03 | 0.52 | 9.25 | 41.9 | 191 |
| Layer2.1.conv1 | 9.2e4 | 0.05 | 0.81 | 14.0 | 65.1 | 309 |
| Layer3.1.conv1 | 3.7e5 | 0.29 | 4.83 | 84.8 | 392 | 2056 |

Uniform quantization defined in Eq. 1 and Eq. 2 linearly maps full-precision weights $\boldsymbol{w}^r$ into quantized representations $\boldsymbol{w}^q$. To remove the outlier of weight distribution, clamp operation are used to restrict the real value in the range $[\boldsymbol{w}_l, \boldsymbol{w}_u]$. The quantized output $\boldsymbol{w}^q$ from uniform quantization scheme at $\boldsymbol{b}$-bit with clamp can be defined as:

$$\boldsymbol{w}^q = \boldsymbol{Q}_u(\boldsymbol{w}^r; b, \boldsymbol{w}_l, \boldsymbol{w}_u) = \boldsymbol{s} \times \text{round}[\text{clamp}(\boldsymbol{w}^r; \boldsymbol{w}_l, \boldsymbol{w}_u)/\boldsymbol{s}], \tag{11}$$

where $[\boldsymbol{w}_l, \boldsymbol{w}_u]$ is the quantization range, $\boldsymbol{\Delta} = \boldsymbol{w}_u - \boldsymbol{w}_l$ is the range length, $\boldsymbol{s} = \frac{\Delta}{\boldsymbol{N}-1}$ is the scaling factor, $\boldsymbol{N} = 2^b$ is the total number of quantization bins. Previous work (You, 2010) has proved that the expected quantization error squared for the $\boldsymbol{b}$-bit uniform quantizer is denoted as:

$$\mathbb{E}(\boldsymbol{\Omega}_u^2; b, \boldsymbol{w}_l, \boldsymbol{w}_u) = \frac{\boldsymbol{s}^2}{12} = \boldsymbol{C}(b)\boldsymbol{\Delta}^2, \tag{12}$$

where $\boldsymbol{C}(b) = \frac{1}{12(2^b-1)^2}$ for uniform distributions. This is why we use $\boldsymbol{\lambda}_b = (2^b - 1)^2$ in Eq. 6 to balance between different bitwidths. In our stochastic quantization scheme, we assume the same quantization range $[\boldsymbol{w}_l, \boldsymbol{w}_u]$ are used for neighboring quantization level. the expectation of quantization error at $\boldsymbol{b}_i$-bit width DBP $\boldsymbol{\beta}$ can be computed as:

$$\mathbb{E}(\boldsymbol{\Omega}_s^2; b_i) = \boldsymbol{\beta}_{b_i}\mathbb{E}(\boldsymbol{\Omega}_u^2; b_i) + (1 - \boldsymbol{\beta}_{b_i})\mathbb{E}(\boldsymbol{\Omega}_u^2; b_{i-1}) = [\boldsymbol{\beta}_{b_i}\boldsymbol{C}(b_i) + (1 - \boldsymbol{\beta}_{b_i})\boldsymbol{C}(b_{i-1})]\boldsymbol{\Delta}^2, \tag{13}$$

where $\boldsymbol{C}(b_i) = \frac{1}{12(2^{b_i}-1)^2}$. Accordingly, the coefficient $\boldsymbol{\lambda}_b = (2^{b_i} - 1)^2$ can still be applied to balance the regularization term at neighboring quantization level while amplify the quantization error for lower bit $b_{i-1}$ by $[(2^{b_i} - 1)/(2^{b_{i-1}} - 1)]^2$.

## B. SDQ on Different Granularity

Our methods use the differentiability of DBPs to learn the bitwidth automatically. The advantage of the proposed SDQ is that DBPs can be inserted into different network granularity. Table. 9 compares the model performance, epochs for MPQ strategy generation, and optimization time per epochs of ResNet18 on different granularity.

*Table 9.* Comparison of top-1 accuracy, total epochs for strategy generation, and time per epochs of different granularity for ResNet18 on ImageNet.

| Granularity | Bit-width(W/A) | Top-1 Acc | Epochs | Time/epoch |
|---|---|---|---|---|
| Net | 4/4 | 68.7 | 30 | 47min |
| Block | 3.77/4 | 71.2 | 60 | 47min |
| Layer | 3.75/4 | 71.7 | 60 | 48min |
| Kernel | 3.81/4 | 71.8 | 90 | 58min |

As we expected, while DBPs can be applied in different granularities, the layer-wise MPQ is still the optimal choice. When applying mixed precision to coarse-grained granularity, including the whole network and blocks, the performance of the trained MPQ network will be restricted because the difference of sensitivity within different components in the network is not fully utilized. Although searching the precision for finer-grained granularity such as different kernels will improve the accuracy of the MPQ model, the time cost for the strategy generation will grow significantly. Introducing more trainable parameters in the optimization process will result in more training instability. Moreover, current hardware accelerators rely highly on the parallelism of computing in the neural network. To the best of our knowledge, there are no kernel-wise hardware accelerators that successfully improve the efficiency in the real deployment of deep learning models. Based on these considerations, we focus more on layer-wise quantization when designing experiments.

# C. Training and Experiment Details

## C.1. Training settings and hyper-parameters

When comparing our SDQ with other quantization methods shown in Table. 1 and Table. 2, we use the training setting and hyper-parameters shown in Table. 10. Note that the most hyper-parameters are the same during the MPQ generation and MPQ training phase, while we observe that a larger batch size yields better performance during MPQ training.

*Table 10.* Detailed hyper-parameters and training scheme of different network architectures

| Network | ResNet20 on CIFAR10 | | ResNet18 on ImageNet | | MobileNetV2 on ImageNet | |
|---|---|---|---|---|---|---|
| Phase | MPQ Generation | MPQ Training | MPQ Generation | MPQ Training | MPQ Generation | MPQ Training |
| Epoch | 100 | 200 | 60 | 90 | 60 | 120 |
| Batch Size | 512 | 1024 | 256 | 512 | 256 | 512 |
| Teacher | - | - | ResNet101 | ResNet101 | ResNet101 | ResNet101 |
| Optimizer | SGD | SGD | Adam | Adam | AdamW | AdamW |
| Initial $lr$ | 0.1 | 0.1 | 5e-4 | 1e-3 | 1e-3 | 1e-3 |
| $lr$ scheduler | MultiStepLR | MultiStepLR | Consine | Consine | Consine | Consine |
| Weight decay | 1e-4 | 1e-4 | - | - | - | - |
| Warmup epochs | - | - | - | - | 30 | 30 |
| Random Crop | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Random Flip | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Color jittering | - | - | ✓ | ✓ | - | - |
| $\lambda_Q$ in Eq. 10 | 1e-6 | - | 1e-7 | - | 1e-7 | - |
| $\lambda_E$ in Eq. 6 | - | 5e-2 | - | 1e-2 | - | 1e-2 |
| $\beta_{\text{thres}}$ | 1e-4 | - | 1e-5 | - | 1e-5 | - |
| Bitwidth candidate $\mathcal{B}$ | 1,2,3,4,5,6,7,8 | - | 2,3,4,5,6,7,8 | - | 2,3,4,5,6,7,8 | - |

## C.2. Training dynamics of SDQ models

Fig. 7 illustrates the training loss and evaluation accuracy for our SDQ MobileNetV2 during the MPQ training phase. As shown in Fig. 1(c), quantized models are challenging to train because the latent non-smooth loss landscape makes models difficult to converge with inaccurately estimated gradients. We found that our proposed Entropy-aware bin regularization targeting to preserve more information can stabilize the training. Especially when the learning rate drops during training, our EBR can effectively reduce the jitters in loss and accuracy.

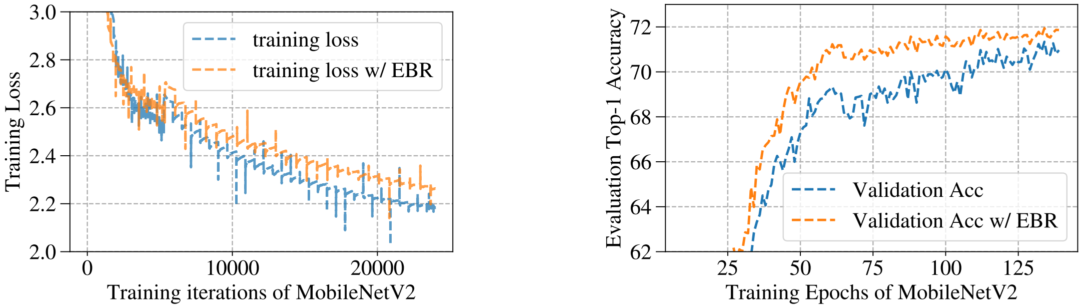


*Figure 7.* Training dynamics of SDQ with and without Entropy-aware bin regularization

### C.3. Detailed Comparison of Quantization Strategy

In Table. 3, we compared our MobileNetV2 quantization strategy with Uhlich et al. (Uhlich et al., 2020) and FracBits (Yang & Jin, 2021) under same initialization and training schemes. The results have proved that our quantization is superior compared to previous strategies. All the details of bitwidth assignment within each layer are shown in Fig. 8. We can see from the figure that all three bitwidth assignments follow similar patterns, such as the bitwidths for the first few layers are higher than average. Generally, layers with more parameters usually have lower precision, but the strategy here yields different bitwidths for layers with a same number of parameters. This supports our statement in Sec. 4.4 that there is no simple heuristic to assign bitwidth based on some specific metrics directly.

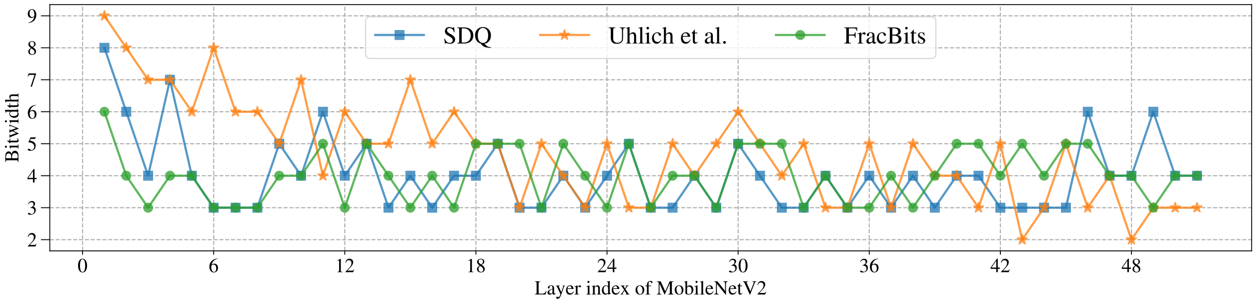


*Figure 8.* Comparison of Quantization Strategy of all layers in MobileNetV2.