From Quantized DNNs to Quantizable DNNs

Kunyuan Du
dukunyuan@sjtu.edu.cn
Shanghai Jiao Tong University, China
Ya Zhang ✉
ya_zhang@sjtu.edu.cn
Haibing Guan
hbguan@sjtu.edu.cn

Abstract

This paper proposes *Quantizable DNNs*, a special type of DNNs that can flexibly quantize its bit-width (denoted as 'bit modes' thereafter) during execution without further re-training. To simultaneously optimize all bit modes, a combinational loss of all bit modes is proposed, which enforces consistent predictions ranging from low-bit mode to 32-bit mode. This *Consistency-based Loss* may also be viewed as certain form of regularization during training. Because outputs of matrix multiplication in different bit modes have different distributions, we further introduce *Bit-Specific Batch Normalization* to reduce conflicts among different bit modes. Experiments on CIFAR100 and ImageNet have shown that compared to quantized DNNs, Quantizable DNNs not only have much better flexibility, but also achieve even higher classification accuracy. Ablation studies further verify that the regularization through the consistency-based loss indeed improves the model's generalization performance. *Source codes will be released in the future.*

1 Introduction

With increasing complexity of Deep Neural Networks (DNNs), great challenges are faced when deploying DNN models to mobile and embedded devices. As a result, model compression and acceleration have received more and more attention in the machine learning community. An important line of research is quantized DNNs, which convert both weights and activations to discrete space. Due to the reduction in bit-width, quantized DNNs have much smaller model size and can be inferenced with high-efficiency fixed-point computation for acceleration. However, when directly quantizing DNNs to 4 bits or less, significant accuracy degradation occurs. To alleviate this problem, quantization-aware training, which simulates the quantization effect with certain bit-width during training and allows the model to adapt to the quantization noise, is widely adopted [5, 8, 22].

In real-world scenarios, different bit-widths may be supported by different devices. For example, Tesla T4 supports 4, 8, 16, and 32 bits, and Watt A1 supports 1, 2, 3, and 4 bits. Either for easily deploying models to different devices, or for dynamic accuracy-efficiency trade-offs on the same device, a quantized DNN that is able to flexibly adjust its bit-width is desirable. However, it is usually non-trivial to re-configure the bit-width of quantized DNNs, because further quantization-aware re-training is generally required in order to guarantee model accuracy. We here propose 'Quantizable' DNNs, a special type of quantized DNNs

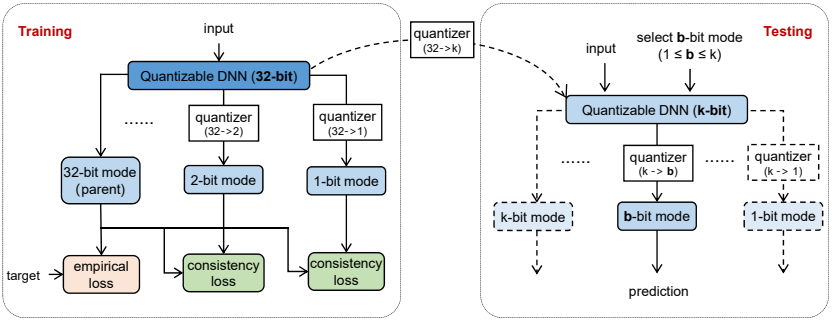

Figure 1: Illustration of Quantizable DNNs. Different bit modes share the same architecture but under different numerical precision.

that can flexibly adjust its bit-width on the fly, i.e. turning on different bit mode by simply applying different quantization precision which aviods quantization-aware re-training. Quantizable DNNs target to pursue a single optimal set of convolutional kernels and fully-connected weights so that different bit modes achieve high accuracy at the same time.

To optimize the Quantizable DNNs, a multi-task framework, which treats the optimization of different bit modes as a set of related sub-tasks, is adopted. Fig. 1 provides an illustration for the Quantizable DNNs. At training, the 32-bit (full-precision) mode serves as the 'parent' model for lower bit modes. For 32-bit mode, the loss function is simply the empirical loss as that of individual quantized DNNs. However, the lower bit modes is known to suffer from noise in gradients [19] and are easily trapped in local minima [25] during training. To optimize the lower bit modes, a *consistency loss*, which encourages different bit modes to produce consistent predictions to that of the 32-bit mode, is further introduced. The consistency loss enforces the lower bit modes to be implicitly guided by the 32-bit mode. Further more, the lower bit modes can be considered as providing a form of regularization to the 32-bit mode through the consistency loss, since the lower bit modes are expected to emphasize on more critical information rather than the redundant details. To some extent, this lower-bit regularization is similar to the well-known 'Dropout' technique [17], by removing less significant bits during training.

Another challenge faced with optimizing the Quantizable DNNs is that outputs of convolutional operation in different bit modes have different distributions, which makes it difficult to properly normalize feature maps from all bit modes with a shared Batch Normalizations [6]. Inspired by [11, 21], we introduce *Bit-Specific Batch Normalization*, which assigns a separate Batch Normalization to each bit mode, to alleviate this problem. The Bit-Specific Batch Normalization introduces negligible additional weights and guarantee the Quantizable DNNs to have approximately the same number of parameters as quantized DNNs.

To validate the effectiveness of Quantizable DNNs, we experiment with two widely used benchmark data sets, Cifar100 [9] and ImageNet [2]. Compared with quantized DNNs, Quantizable DNNs not only enable the on-the-fly dynamic adjustment of mode bit-widths, but also achieve higher classification accuracy with the mutual regularization among different bit-widths. The main contributions of this paper are summarized as follows.

- We design the Quantizable DNN, which is the first DNN model that dynamically adjusts its bit-widths on the fly without quantization-aware re-training.
- We propose a multi-task co-regularization framework, where the lower bit modes and

the 32 bit mode mutually promote each other via *consistency loss* during training.

- We propose the *Bit-Specific Batch Normalization* to alleviate the distribution difference among different bit modes, so that the same parent model may be shared.

2 Related Work

2.1 Quantized DNNs

For smaller model size and higher computational efficiency, both weights and activations of quantized DNNs lie in discrete spaces. Considering whether or not a method needs further re-training, it can be divided into post quantization [2, 13, 23] and quantization-aware training quantization [5, 12, 18, 22, 24]. Most post quantization methods are limited to 8-bit values, and significant performance degradation occurs for ≤ 4 bit quantization. To solve this problem, the widely-used quantization-aware training [5] is applied, which considers quantization noise during training. However, such training process is bit-specific, and converged quantized DNN cannot directly switch to other bit-widths.

In this paper, Quantizable DNNs are implemented based on quantized DNNs, thus careful selection for the base model is needed. We choose Dorefa-net [24] for the following reasons. Firstly, unlike [22], it adopts uniform quantization scheme, which makes it much easier to deploy in various embedded products, e.g. Megvii. Secondly, it is applicable to common network architectures, while [12, 18] requires specially-designed structure. In Dorefa-Net, the k-bit quantizer is defined as Eq.(1), which maps a real number (32-bit) $r \in [0,1]$ to a b-bit discrete value $q \in \{\frac{i}{2^b-1} | \ 0 \leq i \leq 2^b-1, i \in N\}$. The gradient $\frac{\partial q}{\partial r}$ is approximated as 1 [5].

$$q = Q_{32\to b}(r) = \frac{1}{2^b-1}\ round((2^b-1)\ r). \tag{1}$$

2.2 Dynamic Inference

Dynamic inference is the technique to flexibly adjust the network structure during inference to satisfy requirements of computing resources or different tasks. Models allowing dynamic inference can be viewed as an integration of a bunch of sub-DNNs. According to the dimension along which to integrate, existing integration models can be divided into three classes. Firstly, Slimmable DNNs [20, 21] are a type of dynamic DNN that can execute at different channel widths, which can instantly adjust their memory footprint during inference. Then, Mult-Exit DNNs [10, 15] attach multiple classifiers to network structure at different layers, and can decide the model depth to make predictions for fast inference. Different from dynamic models mentioned above, Superposition [1] integrates multiple sub-DNNs that are targeted at different tasks, each sub-DNNs can be retrieved during inference.

3 Quantizable DNNs

3.1 Problem Statement

We first formally define Quantizable DNN under supervised learning setting. Given a training data set $S_t = \{(x_i, y_i)\}_{i=1}^N$, where $\{x_i\}_{i=1}^N$ are the input variables and $\{y_i\}_{i=1}^N$ are the corresponding target variables, a k-bit Quantizable DNN is trained as a special type of Quantized DNN $\hat{y} = \mathcal{F}_k(x; W_k)$, where $\hat{y}$ is the prediction for the corresponding target variable, k is

the model's bit-width, W_k is the quantized model weight in k-bit, x is the input of one data sample. Different from existing k-bit quantized DNN which can only run with the fixed bit width and requires further re-training to change its bit-width at run time, the Quantized DNN can flexibly adjusts its bit-width, i.e. bit mode, on the fly. Given a desired bit mode $b(1 \leq b \leq k)$ at run-time, the k-bit Quantizable DNN can be switched to b-bit mode with $\mathcal{F}_b(x;W_b) = Q(\mathcal{F}_k, W_k, b)(x)$, where Q is a pre-defined quantizer. For Quantizable DNN, we want to simultaneously maximize the accuracy of $\mathcal{F}_b(x;W_b), \forall b \in (1 \leq b \leq k)$.

3.2 Overall Framework

We denote $\mathcal{F}_k(x;W_k)$ as the k-bit Quantizable DNN. During training, a 32-bit mode is trained, which serves as the 'parent' model for other bit modes, i.e. $mode\ list_{train} = \{1,2,...,k,32\}$. The 'parent' model is jointly optimized by all bit modes in *mode list* under multi-task frameworks. For a mini-batch of training data, the Quantizable DNN conducts forward and backward computations in each bit mode and accumulates the gradient. Weights are updated after traversing all bit modes. In the following section, we first introduce the *Consistency-based Training Objective* to optimize the model, which enforces lower bit modes to produce consistent performance with 32-bit mode. To resolve the conflicts between different bit modes, we propose *Bit-Specific Batch Normalization* to normalize outputs in different modes with corresponding learnable affine functions, which is crucial for the performance of Quantizable DNNs. For smaller model size, the 'parent' model can be discarded after training, and the k-bit Quantizable DNN can be directly retrieved from the 32-bit 'parent' model with pre-defined k-bit quantizer $Q_k(\cdot)$, i.e. $\mathcal{F}_k(x;W_k) = Q_{32\to k}(\mathcal{F}_{32}, W_{32})(x)$. In this case, to switch to b-bit mode from $\mathcal{F}_k(x;W_k)$, the quantizer should satisfy $Q_{k\to b}(\mathcal{F}_k, W_k)(x) = Q_{32\to b}(\mathcal{F}_{32}, W_{32})(x)$, which is crucial to ensures the consistency between training and testing. To achieve this, we re-design the pre-defined quantizer via *Thresholds Alignment* trick.

3.3 Consistency-based Training Objective

In this section, we introduce the overall optimization objective of the Quantizable DNN. Since 32-bit mode is introduced as the 'parent' model for training, we denote the Quantizable DNN as $\mathcal{F}_{32}(x, W_{32})$ accordingly. Training a Quantizable DNN can be formulated as a multi-task problem, where each bit mode is treated as a sub-task. The sub-loss function attached to i-bit mode is denoted as $\mathcal{L}_i$. Integrating these sub-loss functions, we can obtain the overall training objective $\mathcal{L}_{all}$:

$$\mathcal{L}_{all} = \sum_{k \in mode\ list} \alpha_k \mathcal{L}_k + \gamma ||Q||^2, \tag{2}$$

where α_k denotes pre-defined weights, and γ is the balancing hyper-parameter between empirical loss and regularization, $||Q||^2$ denotes weight decay. A higher value of α_k encourages the Quantizable DNN to put more attention on k bit mode. In this paper, we treat each bit mode equally and set $\alpha_k = 1$ for all k. In classification tasks, for $k = 32$, $\mathcal{L}_k$ is simply the widely-adopted cross-entropy loss supervised by ground-truth y. However, for $k < 32$, we instead introduce consistency loss to ensure their performance, which utilizes predictions of 32 bit mode $\mathcal{F}_{32}(x, W_{32})$ as supervision for $\mathcal{F}_k(x, W_k)$. Note that $\mathcal{F}_k(x, W_k)$ can be directly obtained from $\mathcal{F}_{32}(x, W_{32})$. Such strategy is adopted because there should be certain internal consistency between predictions of different bit modes since they are integrated into a unified structure. The consistency loss is expressed as Eq.(3):

$$\mathcal{L}_{k(k<32)} = KL(\sigma(\frac{\mathcal{F}_{32}(x, W_{32})}{T}),\ \sigma(\frac{\mathcal{F}_k(x, W_k)}{T})), \tag{3}$$

where $KL(\cdot)$ and $\sigma(\cdot)$ denote Kullback-Leibler divergence and softmax function respectively. Inspired by Knowledge Distillation [4], we introduce a hyper-parameter T to control the smoothness of the supervision, which can explore the 'dark' knowledge between classes. Note that gradients from consistency loss to $\mathcal{F}_{32}(x, W_{32})$ are ignored.

3.4 Bit-specific Batch Normalization

Batch Normalization [6] is proposed to normalize the channel-wise features y with a set of parameters $(\gamma, \beta, \mu, \sigma)$:

$$BN(y) = \gamma \frac{y - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \tag{4}$$

where γ and β are learnable parameters, ε is a small value which can be neglected. μ and σ^2 are means and variances of channel-wise features. Batch Normalization is crucial for quantized DNNs, which maps activations to approximate Gaussian distributions $\mathcal{N}(0,1)$ to make most values lie in quantization interval. However, for Quantizable DNNs, since matrix multiplication outputs produced by different bit modes have different distributions, shared Batch Normalization cannot properly normalize outputs for all bit modes.

Instead of sharing Batch Normalization layers, we propose *Bit-specific Batch Normalization*, which has two variants. For **variant A**, we assign private (μ_k, σ_k) for each bit mode. When training or inference in different modes, channel-wise feature maps can be mapped to the same distribution with respective (μ_k, σ_k) for further quantization. Formally, we denote $y_k = conv(w_k, a_k)$, where a_k represents the k-bit activations from the previous layer and w_k is the k-bit weights in the current convolution layer. *conv* denotes convolution operation and y_k is its output. Note that the output y_k is not limited to k-bit. The **variant A** of *Bit-specific Batch Normalization* can be defined as:

$$BSBN^A(y_k) = BSBN^A_k(y_k) = \gamma \frac{y_k - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + \beta, \tag{5}$$

where γ and β are shared learnable parameters. μ_k and σ_k^2 are private statistical parameter for k-bit mode, which can be either updated as other parameters or directly estimated during inference via post-training strategy [20]. With such strategy, **variant A** can introduce no additional parameters and enable Quantizable DNNs to achieve usable performance. Based on **variant A**, we further introduce a **variant B**, which assigns not only private (μ_k, σ_k), but also private (γ_k, β_k) to each mode:

$$BSBN^B_k(y_k) = \gamma_k \frac{y_k - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + \beta_k. \tag{6}$$

The **variant B** can bring more flexibility to each bit mode and further ease their conflicts. For example, channels that have texture-rich information in higher bit modes may convey negligible information in low-bit modes, and the latter can assign a lower value to the corresponding γ_k to reduce their interference. Though **variant B** introduces additional parameters, the cost can be neglected, because the parameters in Batch Normalization are usually less than 2% of the total model. And it has no effect on inference speed, since in each mode, only the corresponding normalization operation is included in inference graph. **Variant B** is adopted in our experiments unless otherwise stated.

3.5 Quantizer Re-design by Thresholds Alignment

Equipped with components introduced above, the b-bit mode can be directly retrieved from the trained 32-bit 'parent' model via pre-defined quantizer, $\mathcal{F}_b(x;W_b) = Q_{32\to b}(\mathcal{F}_{32},W_{32})(x)$. However, it may fail to be retrieved from a k-bit ($b<k<32$) Quantizable DNN, because the pre-defined quantizer, e.g. Eq.(1), may not satisfy $Q_{k\to b}(\mathcal{F}_k,W_k)(x) = Q_{32\to b}(\mathcal{F}_{32},W_{32})(x)$. To solve this problem, we re-design the pre-defined quantizer with 'thresholds alignment' constraint, i.e. the quantization thresholds of $Q_{32\to b}$ are forced to a subset of $Q_{32\to k}$. The necessity of this constraint is proved in Section 1 of Supplementary material. With 'thresholds alignment' trick, we re-design the quantizer Eq.(1) to Eq.(7):

$$Q_b(r) = \begin{cases} Q_{32\to b}(r) = \frac{1}{2^b-1} clamp\ (round\ (2^b r - 0.5), 0, 2^b-1), & r\ is\ 32\ bit, \\ Q_{k\to b}(r) = \frac{1}{2^b-1} round\left(\frac{2^k-1}{2^{k-b}} r - 0.5\right), & r\ is\ k(k<32)\ bit. \end{cases} \tag{7}$$

With Eq.(7), we can safely discard the 32-bit 'parent' model and only store the k-bit Quantizable DNN in testing phase for small model size. Note that the k-bit Quantizable DNN can switch to exactly the same b-bit ($b<k$) mode with that derived from 'parent'.

4 Experiments

4.1 Implementation Details

To validate the performance of Quantizable DNNs, we compare it with individual quantized DNNs on Cifar100 [9] and ImageNet [2] datasets, in terms of classification accuracy. Our implementation are based on PyTorch [14]. Cifar100 has 40,000 training images, 10,000 validation images and 10,000 test images. Note that since there is no official split, we divide training/validation set by ourselves. ImageNet has 1,280,000 training images and 50,000 validation images. Results on CIFAR100 are average of 3 runs. All '$\pm$' in tables denotes standard deviation. To ensure fairness, Both Quantizable DNNs and corresponding quantized DNNs are trained from scratch for the same epochs, with the same batch size and learning rate. Common data augmentation techniques, e.g. Random Resized Crop and Random Horizontal Flip are adopted for both models. The hyper-parameter T for *consistency loss* is empirically set to 2 for all experiments, which is selected based on the validation set of CIFAR100.

4.2 Classification Performance

Table 1 and Table 2 provide the results on CIFAR100 and ImageNet, respectively. On CIFAR100, we experiment with a Resnet variant which removes the first pooling layer due to the small image size (32×32). All models in Table 1 are trained for 100 epochs with the batch size of 128. On ImageNet, Quantizable DNNs are implemented based on standard AlexNet and Resnet-18. All experiments are trained for 45 epochs with a batch size of 256.

It can be seen that a single Quantizable DNN can even achieve higher overall classification accuracy than a bunch of individual quantized DNNs. On CIFAR100, Low bit modes in Quantizable DNNs outperforms quantized DNNs by 2.81% to 3.29% . And 32-bit mode achieves 0.71% gains due to the regularization from lower bit modes. Similar results are also observed for Alexnet on ImageNet. The accuracy gains verify the effectiveness of the co-regularization scheme in Quantizable DNNs. When experiment with Resnet-18 on ImageNet, the performance of 32-bit mode is degraded by -1.33%, we attribute it to its compact

Model	Top1 Acc(%)	Model	Top1 Acc(%)
32-bit Resnet-Cifar	70.66 ± 0.16		**71.37 ± 0.25**
4-bit Quantized Resnet-Cifar	68.26 ± 0.12		**71.25 ± 0.17**
3-bit Quantized Resnet-Cifar	67.87 ± 0.19	Quantizable Resnet-Cifar	**71.16 ± 0.29**
2-bit Quantized Resnet-Cifar	67.69 ± 0.26		**70.50 ± 0.59**
1-bit Quantized Resnet-Cifar	61.92 ± 0.30		**64.95 ± 0.16**

Table 1: Top-1 test accuracy on CIFAR100.

Model	Top1 Acc(%)	Model	Top1 Acc(%)
32-bit Alexnet	61.38%		**62.86%**
4-bit Quantized Alexnet	60.67%		**61.68%**
3-bit Quantized Alexnet	58.88%	Quantizable Alexnet	**60.76%**
2-bit Quantized Alexnet	52.58%		**56.66%**
1-bit Quantized Alexnet	38.97%		**39.98%**
32-bit Resnet-18	**68.60%**		67.27%
4-bit Quantized Resnet-18	65.93%		**66.94%**
3-bit Quantized Resnet-18	65.03%	Quantizable Resnet-18	**66.28%**
2-bit Quantized Resnet-18	61.73%		**62.91%**
1-bit Quantized Resnet-18*	**50.67%**		-

Table 2: Top-1 validation accuracy on ImageNet.

network architecture, which is more likely to be over-regularized. Note that 1-bit mode is not included in mode list for Quantizable Resnet-18 due to its unique incompatibility, which will be further explained in **Section 4.3**.

4.3 Analysis of Aggressive 1-Bit Mode

When conducting experiments under (ImageNet, Resnet-18) setting, we observe that the performance of 1-bit mode is significantly worse than 1-bit quantized Resnet-18 (by $\approx -7\%$). In response to this phenomenon, we conduct a special analysis for 1-bit mode in this section. Compared to 2-4 bit modes, the most notable feature of 1-bit mode is the mutation of distribution characteristics, which is demonstrated in Fig. 2. When quantized to 2-4 bit, the quantized weights still hold a Gaussian-like (bell-shaped) distribution. However, when further quantized to 1-bit, weights then turn into Bernoulli distribution, which may make it difficult for 1-bit mode to be compatible with other bit modes when integrated into a unified model.

However, for experiments under (ImageNet, Alexnet) and (CIFAR100, Resnet) settings, no degradation is observed for 1-bit mode. We conjecture it is the redundant capacity that allows the model to tolerate the incompatibility from 1-bit mode, because Alexnet (239M) has much more parameters than Resnet-18 (46M) and the task of CIFAR100 is much easier than ImageNet. To validate this hypothesis, we conduct experiments on CIFAR100 using Quantizable Resnet-Cifar with different channel numbers ($1\times, 0.75\times, 0.5\times, 0.25\times$). The fewer channels there are, the less redundancy there is in the model. Fig. 3 shows the accuracy gains compared with quantized DNNs. It can be seen that as the channel number decreases, performance gains of 1-bit mode decreases rapidly compared with 2-4 bit modes. This phenomenon reveals that the more compact the model is, the more obvious the incompatibility

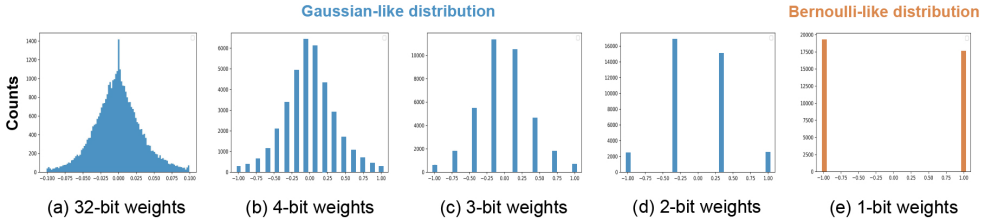

Figure 2: Visualization of weight distributions in different bit modes. 1 bit mode has fundamental difference with other modes.

from 1-bit mode is. On this basis, we further speculate that 1-bit mode can bring negative impacts on other bit modes when integrated in a compact model, and Table 3 verifies our speculation. Therefore, the 1-bit mode is discarded for Quantizable Resnet-18 (Table 2).

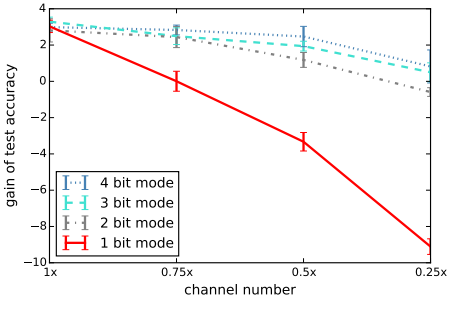

Figure 3: Gain of test accuracy for different bit modes on models with different channel numbers.

Mode	Model A (%)	Model B (%)
32-bit	61.88 ± 0.77	$\mathbf{63.73 \pm 0.34}$
4-bit	62.03 ± 0.67	$\mathbf{63.60 \pm 0.41}$
3-bit	61.32 ± 0.44	$\mathbf{63.01 \pm 0.24}$
2-bit	57.99 ± 0.13	$\mathbf{58.34 \pm 0.10}$
1-bit	37.37 ± 0.22	-

Table 3: Impacts of 1-bit mode on other bit modes. Model A and Model B are Quantizable Resnet-Cifar ($0.25\times$) with and without 1-bit mode respectively.

4.4 Ablation Study

We propose *Bit-Specific Batch Normalization* to enable Quantizable DNNs to converge, and optimize it with *Consistency-based Training Objective*. To verify the effectiveness of these two components, ablation study is conducted on CIFAR100.

4.4.1 Effectiveness of Bit-specific BN

In this section, we make a comparision between Batch Normalization [6], *Bit-Specific Batch Normalization* varaint A and varaint B. Results are presented in Table 4. Since BN fails to resolve the conflict between different bit modes, it has poor overall performance as expected. For BSBN^A, since we normalize feature maps from different bit modes with private (μ_k, σ_k), significant improvement is observed for all bit modes. It futher verifies that the differences in output distribution is the main conflict among different bit modes. On this basis, BSBN^B further assigns more flexibility to each bit mode via private $(\gamma_k, \beta_k, \mu_k, \sigma_k)$ and brings more performance gains to the model. In different situations, we can choose to use a particular variant as needed.

Mode	BN(%)	BSBN^A(%)	BSBN^B(%)
32-bit	1.10 ± 0.17	70.04 ± 0.32	$\mathbf{71.37 \pm 0.25}$
4-bit	6.36 ± 1.59	69.73 ± 0.37	$\mathbf{71.25 \pm 0.17}$
3-bit	20.99 ± 3.45	69.84 ± 0.35	$\mathbf{71.16 \pm 0.29}$
2-bit	62.38 ± 0.68	69.39 ± 0.54	$\mathbf{70.50 \pm 0.59}$
1-bit	2.95 ± 0.28	64.54 ± 0.11	$\mathbf{64.95 \pm 0.16}$

Table 4: Ablation study for *Bit-Specific Batch Normalization*(BSBN).

Mode	Cross-entropy(%)	Consistency loss(%)
32-bit	70.02 ± 0.29	$\mathbf{71.37 \pm 0.25}$
4-bit	69.75 ± 0.31	$\mathbf{71.25 \pm 0.17}$
3-bit	69.63 ± 0.07	$\mathbf{71.16 \pm 0.29}$
2-bit	68.50 ± 0.18	$\mathbf{70.50 \pm 0.59}$
1-bit	61.01 ± 0.23	$\mathbf{64.95 \pm 0.16}$

Table 5: Ablation study for *Consistency-based Training Objective*.

4.4.2 Effectiveness of Training Objective

During training, each low bit mode is attached with a consistency loss rather than widely-used cross-entropy loss. The consistency loss enforces Quantizable DNNs to produce consistent predictions when degraded to low bit modes. Comparison between these two loss functions is presented in Table 5, and the consistency loss brings significant accuracy gains for all bit modes.

5 Efficiency Analysis

A Quantizable DNN can be viewed as the integration of multiple quantized DNNs. In this section we make a further comparision between a set of quantized DNNs and the corresponding Quantizable DNN from the following aspects.

Training time The time occupied by data reading and augmentation cannot be ignored during training. For Quantizable DNNs, less time is taken for training since all bit modes share the same training data. In our implementation (GTX 1080 Ti, 20 Cpu cores), Quantizable Alexnet consumes $0.9\times$ training time per epoch than individually trained quantized DNNs.

Memory footprint & Speed Each bit mode in Quantizable DNNs is identity to a quantized DNN, therefore both models can inference with the same memory footprint and high speed, e.g. 1-bit model can achieve $58\times$ acceleration [16].

Model size For m quantized DNNs with different bit-widths $\{Q_1, \cdots, Q_m\}$, the total model size is $\sum_{i=1}^{m} \textit{model size}(Q_i)$, while the model size of the Quantizable DNN is approximately $\max_i \textit{model size}(Q_i)$, which is only 40% of the former when $\textit{bit modes} = \{1,2,3,4\}$.

Switch time To switch to different bit-widths with quantized DNNs, extra time is needed to re-load the new model, which is determined by the specific hardware system. However, Quantizable DNNs can turn on different bit modes on the fly and achieve instant accuracy-efficiency trade-off.

6 Conclusion

We propose the first DNN model that can adjust its bit-width on the fly, namely Quantizable DNNs. Compared with quantized DNNs, Quantizable DNNs can be instantly converted to different bit modes as needed, which provides much more flexibility in real application scenarios. Besides, the proposed model can even achieve higher accuracy than individual

quantized DNN due to the co-regularization effects between 32-bit mode and low-bit mode. In the future, Quantizable DNNs can be combined with AutoML [3] to efficiently explore optimal bit-width for different layers.

Acknowledgement

This work is supported by the National Key Research and Development Program of China (No. 2019YFB1804304), SHEITC (No. 2018-RGZN-02046), 111 plan (No. BP0719010), and STCSM (No. 18DZ2270700), and State Key Laboratory of UHD Video and Audio Production and Presentation. The computations in this paper were run on the π 2.0 cluster supported by the Center for High Performance Computing at Shanghai Jiao Tong University.

References

[1] Brian Cheung, Alexander Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno Olshausen. Superposition of many models into onef. In *Advances in Neural Information Processing Systems*, pages 10867–10876, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[8] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[10] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1891–1900, 2019.

[11] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[12] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[13] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1325–1334, 2019.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[15] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1355–1364, 2019.

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[18] Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 568–577, 2019.

[19] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

[20] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1803–1811, 2019.

[21] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[22] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[23] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International Conference on Machine Learning*, pages 7543–7552, 2019.

[24] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[25] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7920–7928, 2018.