

Effective Training of Convolutional Neural Networks with Low-bitwidth Weights and Activations

Bohan Zhuang, Jing Liu, Mingkui Tan, Lingqiao Liu, Ian Reid, Chunhua Shen

Abstract—This paper tackles the problem of training a deep convolutional neural network of both low-bitwidth weights and activations. Optimizing a low-precision network is very challenging due to the non-differentiability of the quantizer, which may result in substantial accuracy loss. To address this, we propose three practical approaches, including (i) progressive quantization; (ii) stochastic precision; and (iii) joint knowledge distillation to improve the network training. First, for progressive quantization, we propose two schemes to progressively find good local minima. Specifically, we propose to first optimize a net with quantized weights and subsequently quantize activations. This is in contrast to the traditional methods which optimize them simultaneously. Furthermore, we propose a second progressive quantization scheme which gradually decreases the bit-width from high-precision to low-precision during training. Second, to alleviate the excessive training burden due to the multi-round training stages, we further propose a one-stage stochastic precision strategy to randomly sample and quantize sub-networks while keeping other parts in full-precision. Finally, we adopt a novel learning scheme to jointly train a full-precision model alongside the low-precision one. By doing so, the full-precision model provides hints to guide the low-precision model training and significantly improves the performance of the low-precision network. Extensive experiments on various datasets (e.g., CIFAR-100, ImageNet) show the effectiveness of the proposed methods.

Index Terms—Quantized neural network, relaxed optimization, knowledge distillation, image classification.



1 INTRODUCTION

State-of-the-art deep neural networks [1], [2], [3] usually involve millions of parameters and need billions of FLOPs for training and inference. The significant memory consumption and computational cost can make it intractable to deploy models to mobile, embedded hardware devices. To improve the computational and memory efficiency, various solutions have been proposed, including network pruning [4], [5], [6], low rank approximation of weights [7], [8], training a low-precision network [9], [10], [11], [12] and efficient architecture design [13], [14], [15]. In this work, we follow the idea of training a low-precision network and our focus is to improve the training process of such a network. Thus, our work targets the problem of training network with both extremely low-bit weights and activations.

The solutions proposed in this paper contain three components. They can be applied independently or jointly. The first component is the progressive quantization which consists of two schemes. The first strategy is to adopt a two-stage training process. At the first stage, only the weights of a network is quantized. After obtaining a sufficiently good solution of the first stage, the activation of the network is further required to be in low-precision and the network is trained again. Essentially, this two-step approach first solves a related sub-problem, i.e., training a network with only low-bit weights and the solution of the sub-problem provides a good initial point for training our target problem. Following the similar

idea, we propose our second scheme by performing progressive training on the bit-width aspect of the network. Specifically, we incrementally train a serial of networks with the quantization bit-width (precision) gradually decreased from full-precision to the target precision.

However, the above progressive quantization needs several retraining steps which introduces additional training burdens. To solve this problem, we further propose our second component termed stochastic precision to effectively combine these two strategies into only one training stage. Inspired by dropout strategies [16], [17], we randomly select a portion of the model (i.e., layers, blocks) and activations or weights to quantize while keeping other parts full-precision. This way we effectively improve the gradient flow for training quantized neural networks.

The third component is inspired by the recent progress of information distillation [18], [19], [20], [21], [22]. The basic idea of those works is to train a target network alongside another guidance network. For example, the works in [18], [19], [20], [21], [22] propose to train a small student network to mimic the deeper or wider teacher network. They add an additional regularizer by minimizing the difference between student’s and teacher’s posterior probabilities [19] or intermediate feature representations [18], [22]. It is observed that by using the guidance of the teacher model, better performance can be obtained with the student model than directly training the student model on the target problem. Motivated by these observations, we propose to train a full-precision network alongside the target low-precision network. In our work, the student network has the similar topology as that of the teacher network, except that the student network is low-precision while the teacher network keeps full-precision operations. Moreover, in contrast to standard knowledge distillation methods, we allow the teacher network to be jointly optimized with the student network

- B. Zhuang, C. Shen, L. Liu and I. Reid are with Australian Centre for Robotic Vision, The University of Adelaide. C. Shen is the corresponding author.
E-mail: {firstname.lastname}@adelaide.edu.au
- J. Liu and M. Tan are with South China University of Technology.
E-mail: seliujing@mail.scut.edu.cn, tanmingkui@scut.edu.cn

rather than being fixed since we discover that this treatment enables the two networks adjust better to each other. Interestingly, the performance of both the full-precision teacher and the low-precision student can be improved.

The contributions of this paper can be summarized as follows:

- We propose two progressive quantization schemes for tackling the non-differentiability of quantization operations during training. In the first scheme, we propose a two-step training manner, where the weights are first quantized to serve as a good initialization on further quantizing activations. In the second scheme, we progressively reduce the bit-width during training to find better local minima.
- To reduce the extra training burden, we introduce structured stochastic training, leading to an effective, simplified one-stage training approach.
- To our knowledge, we are the first to propose to improve the low-precision network training using knowledge distillation technique where the full-precision teacher and the quantized student are jointly optimized to adapt to each other. We explore different distilling schemes in Sec. 4 and all produce improved accuracy for the low-precision model.
- We conduct extensive experiments under various precision for variants of ResNet architectures on the image classification task.

This paper extends the preliminary conference version [23] in several aspects. 1) Even though the multi-stage progressive quantization in [23] improves the performance clearly, the multiple re-initialization and fine-tuning steps make the training complex and introduce computation overhead. To solve this problem, we propose a one-stage stochastic precision strategy that enjoys the advantage of the multi-stage progressive quantization. 2) We extend the hint-based joint knowledge distillation to posterior-based scheme, which produces a more comprehensive framework. 3) We now conduct extensive experiments on ImageNet over various architectures to formulate strong and comprehensive baselines for future work. We study several schemes which produce low-precision networks using different distilling strategies and provide interesting analysis.

2 RELATED WORK

A few methods have been proposed to compress deep models and accelerate inference during testing. We can roughly summarize them into four main categories: limited numerical precision, low-rank approximation, efficient architecture design and network pruning.

Limited numerical precision. When deploying DNNs to hardware chips like FPGA, network quantization may be one of the a must process for efficient computing and storage. Fixed-point quantization can be divided into uniform and non-uniform strategies. Uniform approaches [12], [23] design quantizers with a constant quantization step. To reduce the quantization error, the non-uniform strategy [24] is a popular choice. Cai et al. [24] propose to explore the statistics of network activations and batch normalization to select the quantization levels. To make quantization more precise, authors of [25], [26] propose to jointly learn the quantizer and model parameters for better accuracy. Binary neural networks (BNNs) [27], [28] constrain both weights and activations into binary values (i.e., +1 or -1), where the

multiply-accumulations can be replaced by the bitwise `xnor(·)` and `popcount(·)` operations, which can bring significant computation benefits to specialized hardware platforms. The development of binary neural networks may be summarized into two categories. On one hand, some works focus on improving the training of BNNs. In [27], scales are introduced to weights and activations in order to improve the accuracy. Moreover, In [29] authors propose to directly minimize the loss w.r.t. the binarized weights. In [30], a shortcut is added for each binary convolution layer to improve the gradient backpropagation. On the other hand, some works have proposed multiple binarizations to approximate full-precision tensors [31], [32], [33], [34] or structures [35]. A common fundamental problem of fixed-point quantization and BNNs is to approximate gradients of the non-differentiable quantizer (e.g., `sign(·)`, `round(·)`). Most works in literature simply employ “pseudo-gradients” according to the straight-through estimator (i.e., STE) [36]. Recently, Christos *et al.* [37] propose to smooth the discrete quantizer by transforming continuous distributions into categorical distributions which is then relaxed using gumbel-softmax for efficient gradient-based optimization. In contrast, we propose three relaxed training algorithms which can be built upon general quantization approaches, including the quantizer smoothing methods.

Low-rank approximation. Among existing works, some methods attempt to approximate low-rank filters in pre-trained networks [7], [8]. In [8], reconstruction error of the nonlinear responses are minimized layer-wisely, with subject to the low-rank constraint to reduce the computational cost. Other seminal works attempt to restrict filters with low-rank constraints during training phrase [38], [39]. To better exploit the structure in kernels, low-rank tensor decomposition approaches are introduced to remove the redundancy in convolutional kernels in pretrained networks [38], [40].

Efficient architecture design. The increasing demand for highly energy efficient neural networks that are deployable to embed hardware devices has motivated the network architecture design. GoogLeNet [41] and SqueezeNet [42] propose to replace 3×3 convolutional filters with 1×1 size, which tremendously increase the depth of the network while decreasing the complexity. ResNet [3] and its variants [43], [44] utilize residual connections to relieve the gradient vanishing problem when training very deep networks. Recently, depthwise separable convolution employed in Xception [45] and MobileNet [14] have been proved to be efficient and effective. Based on it, ShuffleNet [15] generalizes the group convolution and the depthwise separable convolution to achieve state-of-the-art results. Since it is infeasible to manually explore the optimal architecture from the enormous design space, neural architecture search (NAS) has been attracting more and more attention to automatically search for efficient structures. A main category of methods [46], [47], [48], [49] are based on the reinforcement learning framework while others are built on approximated differentiable gradient-based search [13], [50], [51].

Pruning. Network pruning can be categorized into fine-grained pruning [52], channel pruning [4], [5], [53], [54], [55] and block pruning [56], [57]. Han *et al.* [52], [58] popularize pruning by introducing “deep compression” with a three-stage pipeline: pruning, trained quantization and Huffman coding to effectively reduce the memory requirement of CNNs with almost no loss of accuracy. Channel pruning removes both channels and the related filters from the network, which is the most prominent pruning method and it can be well supported by existing deep learning

libraries with little additional effort. The key issue of channel pruning is to evaluate the importance of channels. Li *et al.* [6] propose to measure the importance of channels by calculating the sum of absolute values of weights. Gao *et al.* [59] learn soft attention on channels for removing redundant ones. Liu *et al.* [55] leverage the scaling layers in batch normalization to effectively identify and prune unimportant channels in the network. Authors of [4], [54] use the gradient magnitude as the channel importance metric. Furthermore, recent works [60], [61] employ AutoML which leverages reinforcement learning to automatically search the pruning ratio for each layer.

Knowledge distillation. Knowledge distillation is initially proposed for model compression, where a powerful wide/deep teacher distills knowledge to a narrow/shallow student to improve its performance [18], [19]. In terms of representation of the knowledge to be distilled from the teacher, existing models typically use teacher’s class probabilities [19] and/or feature representation [18], [21]. Knowledge distillation has been widely used in many computer vision tasks. Zhang *et al.* [62] propose to transfer the knowledge learned with optical flow CNN to improve the action recognition performance. Moreover, several works propose to learn efficient object detection [63], [64] and semantic segmentation [65] with distillation. In contrast to previous approaches, we concentrate on improving the performance of the quantized neural network. By adapting the teacher and student altogether, we can steadily improve the performance of the quantized student network and even the full-precision teacher network.

Dropout. Dropout [16], Maxout [66], DropConnect [67] and DropIn [68] are a category of approaches that propose to stochastically drop intermediate nodes or connections during training to prevent the network from overfitting. They perform different types of regularization. Huang *et al.* [17] further propose stochastic depth regularization via randomly dropping a subset of layers during training. Dong *et al.* [69] propose to randomly quantize a portion of weights to low-precision in the incremental training framework [9]. The method in [69] is developed for only quantizing weights of a network. In our method, we develop extension of it by further randomly quantizing a portion of the network, i.e., *layers or blocks as well as activations and weights*. Moreover, we [23] propose two progressive training strategies: quantizing weights and activations in a two-stage manner; progressively decreasing the bit-width from high-precision to low-precision during the course of training. However, the multi-stage strategy sacrifices the training efficiency. In contrast, we improve the progressive quantization into only a single stage. Our study shows that this extended scheme is complementary to the proposed joint knowledge distillation approach.

3 METHODS

In this section, we first describe the progressive quantization schemes in Sec. 3.1. Then we explain the stochastic precision approach in Sec. 3.2. Then we elaborate the joint knowledge distillation in Sec. 3.3.

3.1 Progressive quantization

3.1.1 Two-step optimization

With the straight-through estimator, it is possible to directly optimize the low-precision network. However, the gradient approximation of the quantization function inevitably introduces

Algorithm 1: Two-stage optimization for k -bit quantization

Input: Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$; A K -bit precision model M_{low}^K .

Output: A low-precision deep model M_{low}^k with weights \mathbf{W}_{low} and activations being quantized into k -bit.

- 1 **Stage 1:** Quantize \mathbf{W}_{low} :
- 2 **for** epoch = 1, ..., L **do**
- 3 **for** $i = 1, \dots, N$ **do**
- 4 Randomly sample a mini-batch data;
- 5 Quantize the weights \mathbf{W}_{low} into k -bit by calling some quantization methods with K -bit activations;
- 6 **Stage 2:** Quantize activations:
- 7 Initialize \mathbf{W}_{low} using the converged k -bit weights from **Stage 1** as the starting point;
- 8 **for** epoch = 1, ..., L **do**
- 9 **for** $i = 1, \dots, N$ **do**
- 10 Randomly sample a mini-batch data;
- 11 Quantize the activations into k -bit by calling some quantization methods while keeping the weights to k -bit;

noisy signal for updating network parameters. Strictly speaking, the approximated gradient may not be the right updating direction. Thus, the training process can be more likely to get trapped at a poor local minimal than training a full precision model. Applying the quantization function to both weights and activations further worsens the situation.

To alleviate this training difficulty, we devise a two-stage optimization procedure as follows. At the first stage, we only quantize the weights of the network while setting the activations to be full precision. After the converge (or after certain number of iterations) of this model, we further apply the quantization function on the activations as well and retrain the network. Essentially, the first stage of this method is a related sub-problem of the target one. Compared to the target problem, it is easier to optimize since it only introduces quantization function on weights. Thus, we are more likely to arrive at a good solution for this sub-problem. Then, using it to initialize the target problem may help the network avoid poor local minima which are likely to be encountered if we train the network from scratch.

Let M_{low}^K be the high-precision model with K -bit. We propose to learn a low-precision model M_{low}^k in a two-stage manner with M_{low}^K serving as the initial point, where $k < K$. The detailed algorithm is shown in Algorithm 1.

3.1.2 Progressive precision

The aforementioned two-stage optimization approach suggests the benefits of using a relatively easy-to-optimize problem to find a good initialization. However, separating the quantization of weights and activations is not the only solution to implement the above idea. In this paper, we also propose a second scheme which progressively lowers the bitwidth of the quantization during the course of network training. Specifically, we progressively conduct the quantization from higher precisions to lower precision (e.g., 32-bit \rightarrow 16-bit \rightarrow 4-bit \rightarrow 2-bit). The model of higher precision will be used as the starting point of the relatively lower precision, analogously to annealing.

Algorithm 2: Progressive precision for accurate CNNs with low-precision weights and activations

Input: Training data $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^N$; A pre-trained 32-bit full-precision model M_{full} as baseline; the precision sequence $\{b_1, \dots, b_n\}$ where $b_n < b_{n-1}, \dots, b_2 < b_1 = 32$.

Output: A low-precision deep model $M_{low}^{b_n}$.

- 1 Let $M_{low}^{b_1} = M_{full}$, where $b_1 = 32$;
 - 2 **for** $i = 2, \dots, N$ **do**
 - 3 Let $k = b_i$ and $K = b_{i-1}$;
 - 4 Obtain M_{low}^k by calling some quantization methods with M_{low}^K being the input;
-

Let $\{b_1, \dots, b_n\}$ be a sequence precision, where $b_n < b_{n-1}, \dots, b_2 < b_1$, b_n is the target precision and b_1 is set to 32 by default. The whole progressive optimization procedure is summarized in Algorithm 2.

Let M_{low}^k be the low-precision model with k -bit and M_{full} be the full precision model. In each step, we propose to learn M_{low}^k , with the solution in the $(i-1)$ -th step, denoted by M_{low}^K , serving as the initial point, where $k < K$.

3.2 Stochastic precision

There is still a problem in the proposed two-stage optimization and the progressive precision schemes, which gradually quantizes the network to low-precision in multi-round training stages. However, the efficiency of training is sacrificed due to the sequentially re-initialization and fine-tuning. To solve this problem, we further propose to improve the progressive quantization strategy into a single stage. Inspired by the recent studies that incrementally or stochastically quantize a certain part of the network, we propose to incorporate the stochasticity into the progressive training by employing a stochastic precision (SP) strategy.

The term ‘‘stochastic structure’’ means that we randomly choose a network structural component, namely, layers, blocks, activations or weights to quantize and keep the rest to be full precision. The specific scheme is elaborated as follows.

Suppose that we decompose the low-precision network M_{low} into Z fragments $M_{low} = \{m_1, \dots, m_Z\}$, where m_i can be any structure such as a convolutional layer or a residual block. For each iteration, we intend to partition the fragments into two sets, a low-precision set $G_q = \{m_{q1}, \dots, m_{qN_q}\}$ and a full-precision set $G_r = \{m_{r1}, \dots, m_{rN_r}\}$, which satisfies the condition:

$$G_q \cup G_r = M_{low}, \text{ and } G_q \cap G_r = \emptyset. \quad (1)$$

where N_q and N_r are the number of elements in two sets respectively.

In our method, we randomly partition M_{low} into G_q and G_r . This is implemented by introducing a binary indicator $\mathbf{b} \in \mathbb{R}^N$. We randomly set $\mathbf{b}(i) = 1$ with probability $(1 - \delta)$, and if $\mathbf{b}(i) = 1$ the i -th fragment is quantized and otherwise are kept to be in full precision. We linearly decrease δ to 0 to ensure the whole network being quantized in the end. Note that this procedure implicitly achieves the effect of [9] but without the need of multi-round training.

To further increase the randomness in quantizing m , we can stochastically choose whether to quantize weights or activations or both of them. This can be implemented by randomly sample a

Algorithm 3: Stochastic precision training algorithm.

Input: Training data $\{\mathbf{x}^t, \mathbf{y}^t\}$; low-precision network M_{low} with parameters \mathbf{W}_{low} ; stochastic ratio δ^t and decay rate μ .

Output: Updated parameters \mathbf{W}_{low} ; stochastic ratio δ^{t+1} .

- 1 Partition M_{low} into N fragments $\{f_1, \dots, f_N\}$;
 - 2 **if** $\delta^t > 0$ **then**
 - 3 Obtain the binary indicator matrix \mathbf{B}^t via uniform sampling with probability δ^t ;
 - 4 Partition the network M_{low} into quantized set $\{G_{qwa}, G_{qw}, G_{qa}\}$ and full-precision set G_r according to \mathbf{B}^t ;
 - 5 Obtain the mixed-precision parameter set $\tilde{\mathbf{Q}}^t = \{q(\mathbf{W}_{qwa}), q(\mathbf{W}_{qw}), \mathbf{W}_{qa}, \mathbf{W}_r\}$ accordingly;
 - 6 **else**
 - 7 $\tilde{\mathbf{Q}}^t = q(\mathbf{W}_{low})$;
 - 8 $\tilde{\mathbf{y}}^t = \text{Forward}(\mathbf{x}^t, \tilde{\mathbf{Q}}^t, \mathbf{B}^t)$;
 - 9 Compute the loss $\mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)$;
 - 10 $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{Q}}^t} = \text{Backward}(\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{y}}^t}, \tilde{\mathbf{Q}}^t, \mathbf{B}^t)$;
 - 11 Update parameters \mathbf{W}_{low} using Adam;
 - 12 $\delta^{t+1} = \delta^t - \mu$;
-

binary indicator matrix $\mathbf{B} \in \mathbb{R}^{N \times 2}$, where its first column is used to decide whether to quantize the weights in the corresponding fragment and the second column is used to decide whether to quantize activations respectively.

As a result, G_q can be further partitioned into three subsets $\{G_{qwa}, G_{qw}, G_{qa}\}$, which represents quantizing both weights and activations, only quantizing weights and only quantizing activations, respectively. Thus, SP can share the advantage of the progressive training in Sec. 3.1.1 and Sec. 3.1.2.

Moreover, in Sec. 4.4.1, we will explore the effect of different structure choices of m as well as the extent of randomness to the final performance.

3.3 Joint knowledge distillation on quantization

The third approach proposed in this work here is inspired by the success of using information distillation [18], [19], [20], [21], [22] to train a relatively shallow network. Specifically, these methods usually use a teacher model (usually a pretrained deeper network) to provide guided signal for the shallower network. Following this spirit, we propose to train the low-precision network alongside another guidance network. Unlike the work in [18], [19], [20], [21], [22], the guidance network shares the similar architecture as the target network but is pretrained with full-precision weights and activations.

However, a pre-trained model may not be necessarily optimal or may not be suitable for quantization. As a result, directly using a fixed pretrained model to guide the target network may not produce the best guidance signals. To mitigate this problem, we do not fix the parameters of a pretrained full precision network as in the previous work [70].

By using the guidance training strategy, we assume that there exists some full-precision models with good generalization performance, and an accurate low-precision model can be obtained by directly performing the quantization on those full-precision models. In this sense, the feature maps of the learned low-precision model should be close to that obtained by directly

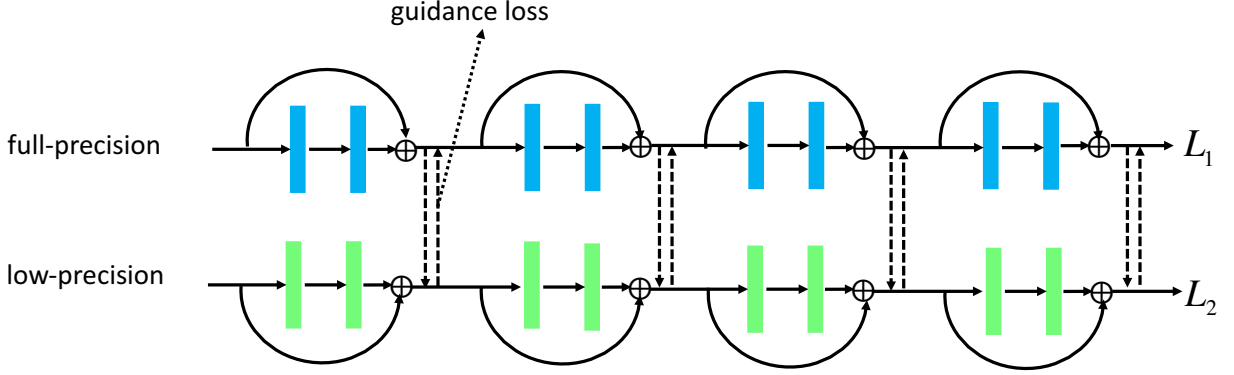


Fig. 1: Demonstration of the guided training strategy. We use the residual network structure for illustration.

performing quantization on the full-precision model. To achieve this, essentially, in our learning scheme, we can jointly train the full-precision and low-precision models. This allows these two models adapt to each other. We even find by doing so the performance of the full-precision model can be slightly improved in some cases.

Formally, let \mathbf{W}_{full} and \mathbf{W}_{low} be the weights of the full-precision model and low-precision model, respectively. Then we employ two distillation strategies: hint-based training and posterior-based training.

Hint-based training Let $\mu(\mathbf{x}; \mathbf{W}_{full})$ and $\nu(\mathbf{x}; \mathbf{W}_{low})$ be the nested feature maps (e.g., activations) of the full-precision model and low-precision model, respectively. To create the guidance signal, we may require that the nested feature maps from the two models should be similar. However, $\mu(\mathbf{x}; \mathbf{W}_{full})$ and $\nu(\mathbf{x}; \mathbf{W}_{low})$ is usually not directly comparable since one is full precision and the other is low-precision. To link these two models, we can directly quantize the weights and activations of the full-precision model. For simplicity, we denote the quantized feature maps by $Q(\mu(\mathbf{x}; \mathbf{W}_{full}))$. Thus, $Q(\mu(\mathbf{x}; \mathbf{W}_{full}))$ and $\nu(\mathbf{x}; \mathbf{W}_{low})$ will become comparable. Then we can define the guidance loss as:

$$R(\mathbf{W}_{full}, \mathbf{W}_{low}) = \frac{1}{2} \| Q(\mu(\mathbf{x}; \mathbf{W}_{full})) - \nu(\mathbf{x}; \mathbf{W}_{low}) \|^2, \quad (2)$$

where $\| \cdot \|$ denotes some proper norms.

Let L_{θ_1} and L_{θ_2} be the cross-entropy classification losses for the full-precision and low-precision model, respectively. The guidance loss will be added to L_{θ_1} and L_{θ_2} , respectively, resulting in two new objectives for the two networks, namely

$$L_1(\mathbf{W}_{full}) = L_{\theta_1} + \lambda R(\mathbf{W}_{full}, \mathbf{W}_{low}), \quad (3)$$

and

$$L_2(\mathbf{W}_{low}) = L_{\theta_2} + \lambda R(\mathbf{W}_{full}, \mathbf{W}_{low}), \quad (4)$$

where λ is balancing hyper-parameter. Here, the guidance loss R can be considered as some regularization on L_{θ_1} and L_{θ_2} .

Posterior-based training Similar to [19], we can also employ the posterior probability as the guidance signal. Let \mathbf{p}_{full} and \mathbf{p}_{low} be the full-precision teacher network and low-precision student network predictions, respectively. To measure the correlation between the two distributions, we employ the KullbackLeibler (KL) divergence:

$$D_{KL}(\mathbf{p}_{full}|\mathbf{p}_{low}) = \sum_{i=1}^N \mathbf{p}_{full}(\mathbf{x}_i) \log \frac{\mathbf{p}_{full}(\mathbf{x}_i)}{\mathbf{p}_{low}(\mathbf{x}_i)} \quad (5)$$

Then the final objectives for the two networks respectively become:

$$L_1(\mathbf{W}_{full}) = L_{\theta_1} + \beta D_{KL}(\mathbf{p}_{full}|\mathbf{p}_{low}), \quad (6)$$

and

$$L_2(\mathbf{W}_{low}) = L_{\theta_2} + \beta D_{KL}(\mathbf{p}_{low}|\mathbf{p}_{full}), \quad (7)$$

where β is the balancing hyper-parameter.

Compared with the hint-based training strategy, the KL divergence based regularizer does not need to select the positions to add the signals, which may be sensitive to the final performance. We empirically compare these two guidance strategies in Sec. 4.2.4.

In the learning procedure, both \mathbf{W}_{full} and \mathbf{W}_{low} will be updated by minimizing $L_1(\mathbf{W}_{full})$ and $L_2(\mathbf{W}_{low})$ separately, using a mini-batch stochastic gradient descent method. The detailed algorithm is shown in Algorithm 4. A high-bit precision model M_{low}^K is used as an initialization of M_{low}^k , where $K > k$. Specifically, for the full-precision model, we have $K = 32$. Relying on M_{full} , the weights and activations of M_{low}^k can be initialized respectively.

Note that the training process of the two networks are different. When updating \mathbf{W}_{low} by minimizing $L_2(\mathbf{W}_{low})$, we use full-precision model as the initialization and apply STE to fine-tune the model. When updating \mathbf{W}_{full} by minimizing $L_1(\mathbf{W}_{full})$, we use conventional forward-backward propagation to fine-tune the model.

Algorithm 4: Guided training with a full-precision network for k -bit quantization

Input: Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$; A pre-trained 32-bit full-precision model M_{full} ; A k -bit precision model M_{low}^k .

Output: A low-precision deep model M_{low}^k with weights and activations being quantized into k bits.

- 1 Initialize M_{low}^k based on M_{full} ;
 - 2 **for** epoch = 1, ..., L **do**
 - 3 **for** $i = 1, \dots, N$ **do**
 - 4 Randomly sample a mini-batch data;
 - 5 Quantize the weights \mathbf{W}_{low} and activations into k -bit by minimizing $L_2(\mathbf{W}_{low})$;
 - 6 Update M_{full} by minimizing $L_1(\mathbf{W}_{full})$;
-

3.4 Remarks on the proposed methods

The proposed three approaches tackle the difficulty in training a low-precision model with different strategies. They can be applied independently. However, it is also possible to combine them together. For example, we can apply the progressive precision to any step in the two-stage approach; we can also apply the joint knowledge distillation to any step in the progressive quantization; we can combine stochastic precision with the joint knowledge distillation approach. Detailed analysis on possible combinations will be empirically evaluated in the experiment section.

4 EXPERIMENTS

To investigate the performance of the proposed methods, we conduct experiments on CIFAR-100 [71] and ImageNet [72]. We employ ResNet [3], PreResNet [44] and AlexNet [1] for experiments. We use a variant of the AlexNet structure by removing dropout layers and add batch normalization after each convolutional layer and fully-connected layer. This structure is widely used in previous works [11], [12].

We explore the effect of the joint knowledge distillation approach in Sec. 4.2, the progressive quantization strategies in Sec. 4.3 and the stochastic precision in Sec. 4.4 in details.

To justify the robustness of the proposed approaches, we conduct experiments on various representative quantization approaches, including uniform fixed-point approach DoReFa-Net [12], non-uniform fixed-point method LQ-Net [25], as well as binary neural network approaches BiReal-Net [30] and Group-Net [35]. We define ‘‘TS’’, ‘‘PP’’, ‘‘SP’’ and ‘‘KD’’ to represent two-step optimization in Sec. 3.1.1, progressive precision in Sec. 3.1.2, stochastic precision in Sec. 3.2 and joint knowledge distillation in Sec. 3.3, respectively.

4.1 Implementation details

As in [12], [23], [24], [27], [35], we quantize the weights and activations of all convolutional layers except that the first and the last layer are kept to be in full-precision. However, we also quantize all the layers so that the model contains complete fixed-point operations. We label this case with a * symbol where we also quantize the input image to 8-bit. In all ImageNet experiments, training images are resized to 256×256 , and a 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted. We do not use any further data augmentation in our implementation. We use a simple single-crop testing for standard evaluation. No bias term is utilized. More details are provided in specific sections. Our implementation is based on Pytorch.

4.2 Effect of the distillation on quantization

To investigate the effect of the joint knowledge distillation approach explained in Sec. 3.3, we explore four different training schemes to obtain a low-precision student network.

4.2.1 Joint fine-tuning of low-precision student and full-precision teacher

In this scheme, both the networks are primed with full-precision pretrained weights as initialization and are jointly optimized. We explore two network structures, including PreResNet and ResNet. When using a certain student network M_{low} , we use the teacher network M_{full} to have either the same or larger depth. The results

are reported in Table 1 and Table 2. For all these cases, we use the posterior-based training strategy as the guidance regularizer. The initial learning rate for M_{low} and M_{full} are set to 0.005 and 0.001, respectively. We train a maximum 30 epochs, and decay the learning rate by 10 at the 15-th and 25-th epoch. We use SGD for optimization, with batch size 256, momentum 0.9 and weight decay $1e-4$. The balancing parameter β is set to 0.5.

Discussion: From the results, we can observe that all our low-precision models surpass the corresponding baselines. It justifies that M_{full} can provide useful auxiliary supervision to assist the convergence of M_{low} . Moreover, the relative improvement with ResNet is larger than that with PreResNet. To highlight, the relative Top-1 improvement w.r.t. 2-bit ResNet-50 is 1.2% while the PreResNet-50 counterpart is 0.5%. This phenomenon can be attributed that quantized ResNet is more difficult to be optimized since the skip connections are also quantized which blocks layers later in the network to access information gained in earlier layers. In this scenario, M_{full} can effectively ease the training of M_{low} by adapting knowledge to each other. We can also justify that keeping the skip connections to high-precision is important to maintain the performance of the low-precision network similar to [30], [73].

Moreover, we can come to an assumption that distillation process becomes more effective when the low-precision network is more difficult to train. This assumption can be further proved by the experiments in Sec. 4.2.2.

In Table 2, we experiment with PreResNet-18 which is paired with various teacher networks but with deeper layers. However, the benefit from using a deeper network saturates at some points. For example, the final trained accuracy of 2-bit PreResNet-18 model paired with PreResNet-50 is 0.3% worse than that obtained by pairing the PreResNet-34 network.

With the simple DoReFa-Net uniform quantization strategy, we can achieve comparable or even higher accuracy compared with the full-precision model using 4-bit precision. It means that we can deploy the 4-bit model in hardware devices with no loss of accuracy which would greatly save memory bandwidth and power consumption.

Interestingly, we also observe that the full-precision teacher can also be improved by learning together with the student. We plot the convergence curves in Figure 2. We can observe that the teacher’s performance drops at the beginning epochs due to inaccurate gradient from the student. During optimization, the student network serves as a regularizer for the teacher network which can even surpass the pretrained baseline.

4.2.2 Learning from scratch vs. fine-tuning

In this scheme, we train a low-precision student from scratch given a pretrained full-precision teacher network. During training, both of the models are mutually updated. The initial learning rates for student and teacher are set to 0.1 and 0.001, respectively. We train a maximum 80 epochs with SGD, and the learning rate is decayed by 10 at 30, 50, 60 and 70 epochs. We adopt the batch size of 256.

The results are reported in Table 3. From the results, we can summarize two instructive statements.

- The relative improvement of KD is more apparent than those that are from fine-tuning. For instance, with 2-bit representations, the relative improvement for PreResNet-50 is 2.2% while the fine-tuning counterpart is 0.5% in Table 2. This is reasonable since learning from scratch

TABLE 1: The accuracy of the quantized ResNet using joint training approach and finetuning. W and A refer to the bitwidth of weights and activations, respectively.

| Precision | | ResNet-18 Baseline | ResNet-18 with ResNet-18 | ResNet-34 Baseline | ResNet-34 with ResNet-34 | ResNet-50 Baseline | ResNet-50 with ResNet-50 |
|-----------|--------|-----------------------|-----------------------------|-----------------------|-----------------------------|-----------------------|-----------------------------|
| 32W, 32A | Top-1% | 69.7 | - | 73.2 | - | 75.6 | - |
| | Top-5% | 89.0 | - | 91.4 | - | 92.2 | - |
| 4W, 4A | Top-1% | 69.4 | 70.0 | 71.3 | 72.9 | 74.5 | 75.3 |
| | Top-5% | 88.9 | 89.9 | 90.0 | 91.3 | 91.5 | 91.7 |
| 2W, 2A | Top-1% | 64.7 | 65.6 | 68.2 | 69.0 | 70.2 | 71.4 |
| | Top-5% | 86.0 | 86.3 | 88.1 | 88.6 | 89.1 | 90.0 |

TABLE 2: The accuracy of the quantized PreResNet using joint training approach and finetuning.

| Precision | | PreResNet-18 Baseline | PreResNet-18 with PreResNet-18 | PreResNet-18 with PreResNet-34 | PreResNet-18 with PreResNet-50 | PreResNet-34 Baseline | PreResNet-34 with PreResNet-34 | PreResNet-50 Baseline | PreResNet-50 with PreResNet-50 |
|-----------|--------|--------------------------|-----------------------------------|-----------------------------------|-----------------------------------|--------------------------|-----------------------------------|--------------------------|-----------------------------------|
| 32W, 32A | Top-1% | 70.0 | - | - | - | 73.5 | - | 76.1 | - |
| | Top-5% | 89.2 | - | - | - | 91.3 | - | 92.8 | - |
| 4W, 4A | Top-1% | 69.8 | 70.0 | - | - | 73.6 | 73.8 | 75.9 | 76.4 |
| | Top-5% | 89.1 | 89.4 | - | - | 91.3 | 91.4 | 92.8 | 93.0 |
| 2W, 2A | Top-1% | 64.5 | 65.2 | 65.4 | 65.3 | 69.3 | 70.0 | 71.2 | 71.7 |
| | Top-5% | 85.9 | 86.2 | 86.4 | 86.4 | 89.0 | 89.4 | 90.1 | 90.5 |

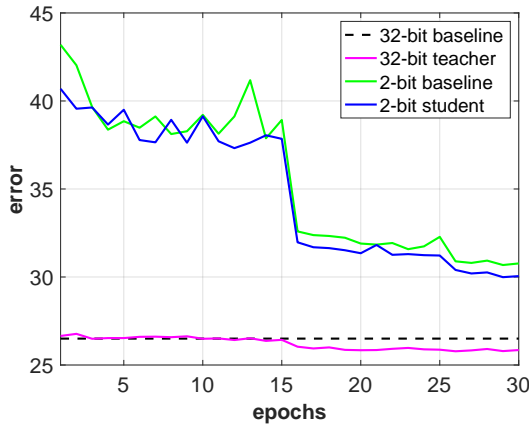


Fig. 2: Both student and teacher are fine-tuned from the pretrained models. We use PreResNet-34 as illustration.

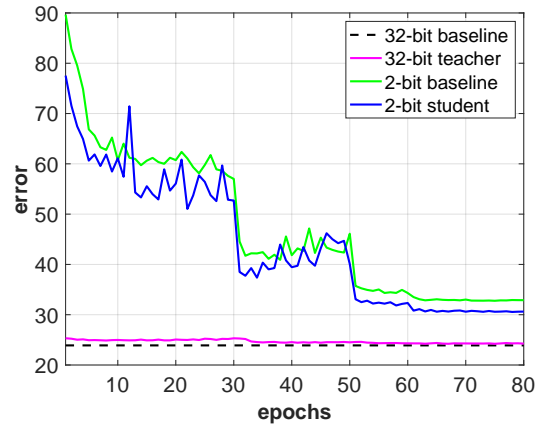


Fig. 3: Student is learnt from scratch while teacher is fine-tuned. PreResNet-50 is used here.

is more challenging than fine-tuning and the auxiliary guidance from the teacher has more affects.

- Fine-tuning performs steadily better than learning from scratch. It shows that the pretrained full-precision model serves as an important initial point.

We also plot the convergence curves in Figure 3.

4.2.3 Learning from the fixed teacher

In this section, we fix the pretrained teacher network and only fine-tune the student network. This is the scheme used by [19] to train their student network. The training details for the student network are the same as those described in Sec. 4.2.1.

From Table 4, we can observe that the improvement is relatively lower than that with pretrained teachers in Table 2. This proves that directly transferring the knowledge from the pretrained teacher may not be optimal or not be suitable for quantization.

Both M_{low} and M_{full} should be jointly optimized to adapt to each other. However, this scheme has an advantage that one can pre-compute and store the guidance signals and access them during training M_{low} , which can save the forward and backward pass computations w.r.t. M_{full} . For better understanding, we

further show the convergence curves for AlexNet* on ImageNet in Figure 4.

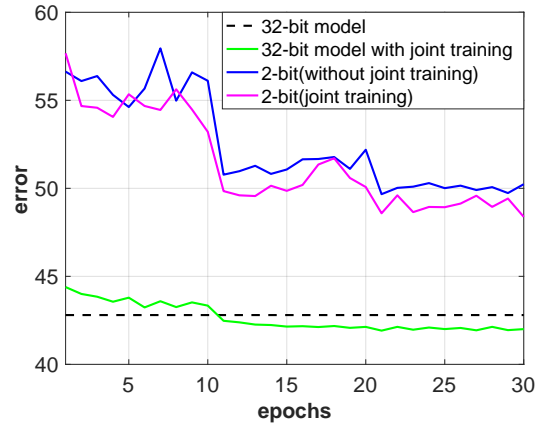


Fig. 4: Joint training vs. fixed teacher using AlexNet* on ImageNet.

4.2.4 Hint-based vs. posterior-based distillation

In this scheme, we further explore the effect of different knowledge distillation strategies as introduced in Sec. 3.3. The results

TABLE 3: The accuracy of the quantized PreResNet using the joint training approach, which is learnt from scratch.

| Precision | | PreResNet-18 Baseline | PreResNet-18 with PreResNet-18 | PreResNet-34 Baseline | PreResNet-34 with PreResNet-34 | PreResNet-50 Baseline | PreResNet-50 with PreResNet-50 |
|-----------|--------|--------------------------|-----------------------------------|--------------------------|-----------------------------------|--------------------------|-----------------------------------|
| 32W, 32A | Top-1% | 70.0 | - | 73.5 | - | 76.1 | - |
| | Top-5% | 89.2 | - | 91.3 | - | 92.8 | - |
| 4W, 4A | Top-1% | 67.9 | 69.2 | 71.5 | 72.9 | 73.9 | 75.4 |
| | Top-5% | 88.2 | 88.7 | 90.1 | 90.9 | 91.6 | 92.6 |
| 2W, 2A | Top-1% | 62.6 | 64.8 | 66.5 | 68.5 | 67.2 | 69.4 |
| | Top-5% | 84.5 | 86.0 | 87.2 | 88.4 | 87.7 | 89.3 |

TABLE 4: The accuracy of the quantized PreResNet using fixed full-precision teacher.

| Precision | | PreResNet-18 Baseline | PreResNet-18 with PreResNet-18 | PreResNet-34 Baseline | PreResNet-34 with PreResNet-34 |
|-----------|--------|--------------------------|-----------------------------------|--------------------------|-----------------------------------|
| 4W, 4A | Top-1% | 69.8 | 70.0 | 73.6 | 73.8 |
| | Top-5% | 89.1 | 89.3 | 91.3 | 91.5 |
| 2W, 2A | Top-1% | 64.5 | 64.9 | 69.3 | 69.7 |
| | Top-5% | 85.9 | 86.2 | 89.0 | 89.2 |

are reported in Table 5. We analyze the sensitivity of the MSE loss position to the final performance. We add the hints at the end of the last block or last two blocks. For hint-based distillation, we set the optimization details the same as those for posterior-based distillation in Sec. 4.2.1 except that the balancing parameter λ is set to 0.1. For hint-based training, we observe that the position of the hints can affect the performance to some extent. For example, adding two hints outperforms the one-hint counterpart by 0.7%.

TABLE 5: Hint-based vs. posterior-based with ResNet-18 on ImageNet.

| Precision | | ResNet-18 Baseline | ResNet-18 posterior | ResNet-18 two hints | ResNet-18 one hint |
|-----------|--------|-----------------------|------------------------|------------------------|-----------------------|
| 2W, 2A | Top-1% | 64.7 | 65.6 | 65.7 | 65.0 |
| | Top-5% | 86.0 | 86.3 | 86.4 | 86.1 |

4.3 Effect of progressive quantization

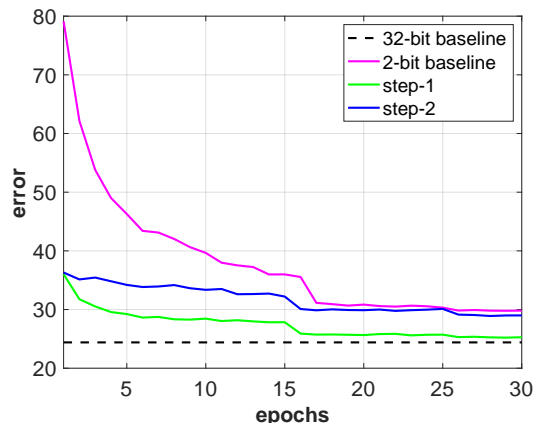
In this part, we explore the effect of the proposed progressive quantization methods.

4.3.1 Effect of the two-step optimization

We further analyze the effect of each stage in the two-step approach in Figure 5. We take the 2-bit ResNet-50 on ImageNet as example. In Figure 5, step-1 has the minimal loss of accuracy. As for the step-2, although it incurs apparent accuracy decrease in comparison with that of the step-1, its accuracy is consistently better than the results of baseline in every epoch. This illustrates that progressively seeking for the local minimum point is crucial for final better convergence, which proves the effectiveness of this simple mechanism.

4.3.2 Effect of the progressive precision strategy

What is more, we also separately explore the progressive precision effect on the final performance. In this experiment, we apply AlexNet and ResNet-50 on the ImageNet dataset. We continuously quantize both weights and activations simultaneously from 32-bit→8-bit→4-bit→2-bit and explicitly illustrate the accuracy change process for each precision in Figure 6. The quantitative results are also reported in Table 6. From the figure we can find that for the 8-bit and 4-bit, the low-bit model has no accuracy

**Fig. 5:** The two-step training approach on ResNet-50.**TABLE 6:** Accuracy (%) of different comparing methods on the ImageNet validation set. All the cases keep skip connections during testing.

| Precision | model | method | top-1 acc. | top-5 acc. |
|-----------|------------|--------------------|-------------|-------------|
| 2W, 2A | ResNet-50 | Baseline | 70.2 | 89.1 |
| | | Baseline + TS | 70.9 | 90.0 |
| | | Baseline + PP | 70.8 | 90.0 |
| | | Baseline + TS + PP | 71.1 | 90.1 |
| 4W, 4A | ResNet-50* | Baseline | 75.1 | 91.8 |
| | | Baseline + TS | 75.3 | 91.9 |
| | | Baseline + PP | 75.4 | 91.8 |
| | | Baseline + TS + PP | 75.5 | 92.0 |
| 2W, 2A | ResNet-50* | Baseline | 67.7 | 86.9 |
| | | Baseline + TS | 69.2 | 87.0 |
| | | Baseline + PP | 68.8 | 86.9 |
| | | Baseline + TS + PP | 69.4 | 87.0 |
| 4W, 4A | AlexNet* | Baseline | 56.2 | 79.4 |
| | | Baseline + TS | 57.7 | 81.0 |
| | | Baseline + PP | 57.5 | 80.8 |
| | | Baseline + TS + PP | 57.8 | 80.8 |
| 2W, 2A | AlexNet* | Baseline | 48.3 | 74.8 |
| | | Baseline + TS | 50.7 | 74.9 |
| | | Baseline + PP | 50.3 | 74.8 |
| | | Baseline + TS + PP | 50.9 | 74.9 |

loss with respect to the full precision model. However, when quantizing from 4-bit to 2-bit, we can observe significant accuracy drop. Despite this, we still observe 2.0% relative improvement by comparing the Top-1 accuracy over the 2-bit baseline, which

proves the effectiveness of the proposed strategy. It is worth noting that the accuracy curves become more unstable when quantizing to lower bit. This phenomenon is reasonable since the precision becomes lower, the quantized value will change more frequently during training.

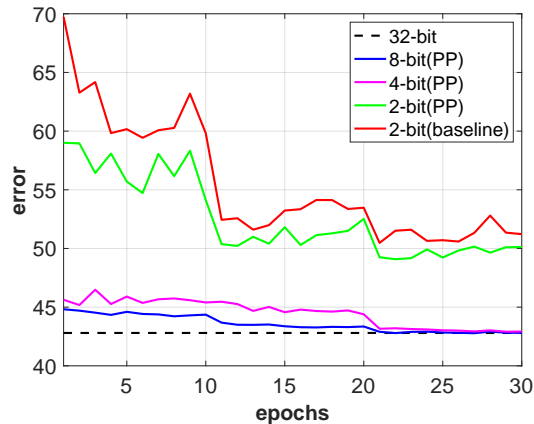


Fig. 6: The progressive training approach on AlexNet*.

4.4 Effect of the stochastic precision

In this subsection, we further explore the effect of the stochastic precision strategy on general quantization approaches. The default structure of the fragment f is a residual block and we stochastically quantize weights and activations in all cases unless special explanations. The results are reported in Table 7. By combining the baseline methods with *SP*, we find apparent performance increase compared with the baselines in all cases. During training, we stochastically keep a portion of network to full-precision and update by the standard gradient-based method. This strategy shares the similar spirit with the progressive quantization to relax the discrete quantizer effectively. Moreover, the proposed stochastic strategy only requires one training stage without fine-tuning the model in many training rounds.

TABLE 7: Accuracy (%) of different comparing methods with SP on the ImageNet validation set.

| model | method | top-1 acc. | top-5 acc. |
|-----------|--------------------|-------------|-------------|
| ResNet-50 | DoReFa-Net (2-bit) | 70.2 | 89.1 |
| | DoReFa-Net + SP | 72.2 | 90.8 |
| ResNet-50 | LQ-Net (3-bit) | 74.2 | 91.6 |
| | LQ-Net + SP | 75.1 | 92.3 |
| ResNet-18 | BiReal-Net | 56.4 | 79.5 |
| | BiReal-Net + SP | 58.8 | 81.2 |
| ResNet-18 | GroupNet (5 bases) | 64.8 | 85.7 |
| | GroupNet + SP | 65.9 | 86.3 |

4.4.1 Effect of different SP policies

We further explore the influence of different choices of the fragment f described in Sec. 3.2 as well as the extent of randomness. We treat GroupNet as our baseline approach and utilize 5 binary bases. The results are reported in Table 8. We explore two different structures of f , including one convolutional layer and one residual block which corresponds to *layerdrop* and *blockdrop* respectively. We further incorporate the randomness of quantizing weights and activations into f and is denoted by *W/A*. From the results, we can find that all the four cases show improved performance compared with the baseline, which justifies adding randomness is a general way for relaxing the low-precision network training.

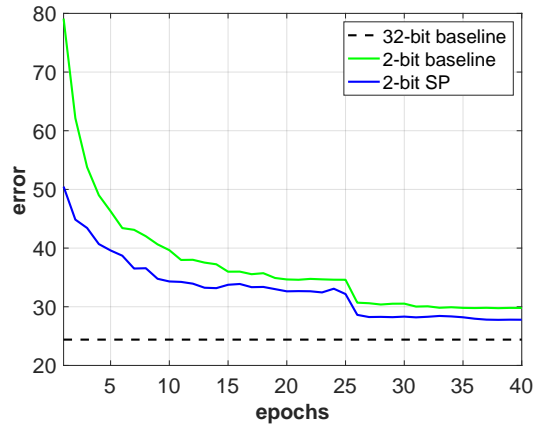


Fig. 7: The stochastic precision training approach on ResNet-50.

By comparing the result of *layerdrop*+*W/A* with *layerdrop*, we can observe performance drop with the increase of randomness. However, *blockdrop*+*W/A* performs slightly better than *blockdrop*. This shows that adding excessive stochasticity can make the gradient updating direction deviate while appropriate extent of randomness can relax the non-differentiable problem to facilitate optimization. Moreover, the accuracy of *layerdrop* and *blockdrop* are very close, which shows that the structure of f is not sensitive to the final performance.

TABLE 8: Accuracy (%) of different stochastic policies on the ImageNet validation set.

| model | method | top-1 acc. | top-5 acc. |
|-----------|----------------------------|-------------|-------------|
| ResNet-18 | GroupNet (5 bases) | 64.8 | 85.7 |
| | GroupNet + blockdrop | 65.6 | 86.3 |
| | GroupNet + layerdrop | 65.7 | 86.5 |
| | GroupNet + blockdrop + W/A | 65.9 | 86.6 |
| | GroupNet + layerdrop + W/A | 65.0 | 86.1 |

4.5 Effect of quantizing all layers

In this part, we further explore the effect of quantizing the first convolution layer and the last classification layer to the final performance. We report the performance in Tables 6 & 11 & 12 & 13. With “2W, 2A”, we can observe that the performance of ResNet-50 beats ResNet-50* by a large margin. This shows that keeping the first and the last layer to high-precision is crucial to preserve the quantized model accuracy. Moreover, the proposed advanced training approaches improves the baseline significantly. For 2-bit precision, the gap between “ResNet-50* TS+PP+KD” and baseline* is 2.3% while “ResNet-50 TS+PP+KD” improves baseline by 1.2%. It further justifies the claim in Sec. 4.2.1 and Sec. 4.2.2 that the proposed training algorithms can be more effective when the model is more challenging to be optimized.

4.6 Combining different training strategies

Finally, we come to our complete approach by combining TS, PP, SP and KD. We first combine TS, PP with KD and the results are shown in Tables 11 & 13. Moreover, we also combine the one-stage SP strategy with KD and the full results are reported in Tables 9 & 10 & 12.

We can observe that the proposed approaches can benefit with each other and further improve the performance on all settings. For instance, with “2W, 2A” in Table 9, we find a 2.4% relative gap

between the baseline on ResNet-50. Even with the basic quantizer in DoReFa-Net, the difference in Top-1 error is only 3%. This strongly justifies that the proposed joint knowledge distillation and the stochastic precision are general training approaches for improving low-bit neural networks.

5 CONCLUSION

In this paper, we have proposed three novel approaches to solve the optimization problem for quantizing the network with both low-precision weights and activations. Firstly, we have proposed the progressive quantization approach which includes two schemes. Specifically, we have proposed a two-step training scheme, where we use the real-valued activations as an intermediate step. We have also observed that continuously quantizing from high-precision to low-precision is also beneficial to the final performance. Moreover, we have proposed a stochastic precision strategy to significantly reduce the training complexity of progressive quantization while still improving the performance.

Furthermore, we have presented to improve the accuracy of low-precision networks with knowledge distillation. In particular, to better take advantage of the knowledge from the full-precision model, we have proposed to jointly learn the low-precision model and its full-precision counterpart. We have explored various distillation schemes and all observed improvements over the baseline. Finally, we have combined the three training approaches to further boost the performance. We have conducted extensive experiments to justify the effectiveness of the proposed approaches on the image classification task.

ACKNOWLEDGMENTS

L. Liu was in part supported by ARC DECRA Fellowship DE170101259. M. Tan was partially supported by National Natural Science Foundation of China (NSFC) 61602185, Program for Guangdong Introducing Innovative and Entrepreneurial Teams 2017ZT07X183.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105. [1, 6](#)
- [2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Repr.*, 2015. [1](#)
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016, pp. 770–778. [1, 2, 6](#)
- [4] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 875–886. [1, 2, 3](#)
- [5] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2, 2017, p. 6. [1, 2](#)
- [6] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017. [1, 3](#)
- [7] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015. [1, 2](#)
- [8] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, 2016. [1, 2](#)
- [9] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *Proc. Int. Conf. Learn. Repr.*, 2017. [1, 3, 4](#)
- [10] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131. [1](#)
- [11] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *Proc. Int. Conf. Learn. Repr.*, 2017. [1, 6](#)
- [12] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016. [1, 2, 6](#)
- [13] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019. [1, 2](#)
- [14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. [1, 2](#)
- [15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018, pp. 6848–6856. [1, 2](#)
- [16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [1, 3](#)
- [17] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 646–661. [1, 3](#)
- [18] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *Proc. Int. Conf. Learn. Repr.*, 2015. [1, 3, 4](#)
- [19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Adv. Neural Inf. Process. Syst. Workshops*, 2014. [1, 3, 4, 5, 7](#)
- [20] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multi-task and transfer reinforcement learning," *Proc. Int. Conf. Learn. Repr.*, 2016. [1, 4](#)
- [21] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Repr.*, 2017. [1, 3, 4](#)
- [22] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662. [1, 4](#)
- [23] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Towards effective low-bitwidth convolutional neural networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. [2, 3, 6](#)
- [24] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5918–5926. [2, 6](#)
- [25] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comp. Vis.*, 2018. [2, 6](#)
- [26] S. Jung, C. Son, S. Lee, J. Son, Y. Kwak, J.-J. Han, and C. Choi, "Joint training of low-precision neural network with quantization interval parameters," *arXiv preprint arXiv:1808.05779*, 2018. [2](#)
- [27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 525–542. [2, 6](#)
- [28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115. [2](#)
- [29] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," in *Proc. Int. Conf. Learn. Repr.*, 2017. [2](#)
- [30] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Proc. Eur. Conf. Comp. Vis.*, 2018. [2, 6](#)
- [31] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 344–352. [2](#)
- [32] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao, "Performance guaranteed network acceleration via high-order residual quantization," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2017, pp. 2584–2592. [2](#)
- [33] Y. Guo, A. Yao, H. Zhao, and Y. Chen, "Network sketching: Exploiting binary structure in deep cnns," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5955–5963. [2](#)
- [34] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Proc. Twenty-Eighth AAAI Conf. on Arti. Intel.*, 2017, pp. 2625–2631. [2](#)
- [35] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural network for accurate image classification and semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. [2, 6](#)

TABLE 9: Accuracy (%) of ResNets on the ImageNet validation set using SP and KD.

| Precision | | ResNet-18 Baseline | ResNet-18 with ResNet-18 | ResNet-34 Baseline | ResNet-34 with ResNet-34 | ResNet-50 Baseline | ResNet-50 with ResNet-50 |
|-----------|--------|-----------------------|-----------------------------|-----------------------|-----------------------------|-----------------------|-----------------------------|
| 32W, 32A | Top-1% | 69.7 | - | 73.2 | - | 75.6 | - |
| | Top-5% | 89.0 | - | 91.4 | - | 92.2 | - |
| 2W, 2A | Top-1% | 64.7 | 65.9 | 68.2 | 70.1 | 70.2 | 72.6 |
| | Top-5% | 86.0 | 87.0 | 88.1 | 89.6 | 89.1 | 90.9 |

TABLE 10: Accuracy (%) of PreResNets on the ImageNet validation set with SP and KD.

| Precision | | PreResNet-18 Baseline | PreResNet-18 with ResNet-18 | PreResNet-34 Baseline | PreResNet-34 with PreResNet-34 | PreResNet-50 Baseline | PreResNet-50 with PreResNet-50 |
|-----------|--------|--------------------------|--------------------------------|--------------------------|-----------------------------------|--------------------------|-----------------------------------|
| 32W, 32A | Top-1% | 70.0 | - | 73.5 | - | 76.1 | - |
| | Top-5% | 89.2 | - | 91.3 | - | 92.8 | - |
| 2W, 2A | Top-1% | 64.5 | 65.7 | 69.3 | 70.3 | 71.2 | 72.3 |
| | Top-5% | 85.9 | 86.6 | 89.0 | 89.5 | 90.1 | 90.7 |

TABLE 11: Accuracy (%) of ResNet-50 and ResNet-50* on the ImageNet with TS, PP and KD.

| Precision | | Baseline* | ResNet-50* TS + PP + KD | Baseline | ResNet-50 TS + PP + KD |
|-----------|--------|-----------|----------------------------|----------|---------------------------|
| 32W, 32A | Top-1% | 75.6 | - | - | - |
| | Top-5% | 92.2 | - | - | - |
| 4W, 4A | Top-1% | 75.1 | 75.7 | 74.5 | 75.3 |
| | Top-5% | 91.9 | 92.0 | 91.5 | 91.7 |
| 2W, 2A | Top-1% | 67.7 | 70.0 | 70.2 | 71.4 |
| | Top-5% | 84.7 | 87.5 | 89.1 | 90.0 |

TABLE 12: Accuracy (%) of AlexNet* on the CIFAR-100 with SP and KD.

| Precision | | AlexNet* Baseline | AlexNet* SP + KD |
|-----------|--------|----------------------|---------------------|
| 32W, 32A | Top-1% | 65.4 | - |
| | Top-5% | 88.3 | - |
| 2W, 2A | Top-1% | 63.9 | 64.9 |
| | Top-5% | 87.6 | 88.1 |

TABLE 13: Accuracy (%) of AlexNet* on the ImageNet with TS, PP and KD.

| Precision | | AlexNet* Baseline | AlexNet* TS + KD | AlexNet* TS + PP + KD |
|-----------|--------|----------------------|---------------------|--------------------------|
| 32W, 32A | Top-1% | 57.2 | - | - |
| | Top-5% | 80.3 | - | - |
| 4W, 4A | Top-1% | 56.8 | 57.8 | 58.0 |
| | Top-5% | 80.0 | 80.9 | 81.1 |
| 2W, 2A | Top-1% | 48.8 | 51.1 | 51.6 |
| | Top-5% | 72.2 | 75.3 | 76.2 |

- [36] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013. **2**
- [37] C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, and M. Welling, "Relaxed quantization for discretized neural networks," in *Proc. Int. Conf. Learn. Repren.*, 2019. **2**
- [38] A. Novikov, D. Podoprikin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 442–450. **2**
- [39] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," in *Proc. Int. Conf. Learn. Repren.*, 2016. **2**
- [40] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277. **2**
- [41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 1–9. **2**
- [42] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016. **2**
- [43] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016. **2**
- [44] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 630–645. **2, 6**
- [45] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 1251–1258. **2**
- [46] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Repren.*, 2017. **2**
- [47] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018. **2**
- [48] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. **2**
- [49] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comp. Vis.*, 2018. **2**
- [50] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Repren.*, 2019. **2**
- [51] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Repren.*, 2019. **2**
- [52] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143. **2**
- [53] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2017, pp. 5058–5066. **2**
- [54] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018, pp. 9194–9203. **2, 3**
- [55] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2017, pp. 2755–2763. **2, 3**
- [56] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comp. Vis.*, 2018. **2**
- [57] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018, pp. 8817–8826. **2**
- [58] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *Proc. Int. Conf. Learn. Repren.*, 2016. **2**
- [59] X. Gao, Y. Zhao, L. Dudziak, R. Mullins, and C.-z. Xu, "Dynamic channel pruning: Feature boosting and suppression," in *Proc. Int. Conf. Learn. Repren.*, 2019. **3**

- [60] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, “N2n learning: Network to network compression via policy gradient reinforcement learning,” in *Proc. Int. Conf. Learn. Reprn.*, 2018. 3
- [61] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “Amc: Automl for model compression and acceleration on mobile devices,” in *Proc. Eur. Conf. Comp. Vis.*, 2018. 3
- [62] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, “Real-time action recognition with enhanced motion vector cnns,” in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016, pp. 2718–2726. 3
- [63] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning efficient object detection models with knowledge distillation,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 742–751. 3
- [64] Y. Wei, X. Pan, H. Qin, W. Ouyang, and J. Yan, “Quantization mimic: Towards very tiny cnn for object detection,” in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 267–283. 3
- [65] T. He, C. Shen, Z. Tian, D. Gong, C. Sun, and Y. Yan, “Knowledge adaptation for efficient semantic segmentation,” *arXiv preprint arXiv:1903.04688*, 2019. 3
- [66] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in *Proc. Int. Conf. Mach. Learn.*, 2013. 3
- [67] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066. 3
- [68] L. N. Smith, E. M. Hand, and T. Doster, “Gradual dropin of layers to train very deep neural networks,” in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016, pp. 4763–4771. 3
- [69] Y. Dong, R. Ni, J. Li, Y. Chen, J. Zhu, and H. Su, “Learning accurate low-bit deep neural networks with stochastic quantization,” in *Proc. Brit. Mach. Vis. Conf.*, 2017. 3
- [70] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018, pp. 4320–4328. 4
- [71] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009. 6
- [72] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *Int. J. Comp. Vis.*, vol. 115, no. 3, pp. 211–252, 2015. 6
- [73] E. Park, D. Kim, S. Yoo, and P. Vajda, “Precision highway for ultra low-precision quantization,” *arXiv preprint arXiv:1812.09818*, 2018. 6