Training Quantized Network with Auxiliary Gradient Module

Bohan Zhuang[1,2] Lingqiao Liu[1] Mingkui Tan[3] Chunhua Shen[1,2*] Ian Reid[1,2]
[1]The University of Adelaide, Australia [2]Australian Centre for Robotic Vision
[3]South China University of Technology, China

March 28, 2019

Abstract

In this paper, we seek to tackle two challenges in training low-precision networks: 1) the notorious difficulty in propagating gradient through a low-precision network due to the non-differentiable quantization function; 2) the requirement of a full-precision realization of skip connections in residual type network architectures. During training, we introduce an auxiliary gradient module which mimics the effect of skip connections to assist the optimization. We then expand the original low-precision network with the full-precision auxiliary gradient module to formulate a mixed-precision residual network and optimize it jointly with the low-precision model using weight sharing and separate batch normalization. This strategy ensures that the gradient back-propagates more easily, thus alleviating a major difficulty in training low-precision networks. Moreover, we find that when training a low-precision plain network with our method, the plain network can achieve performance similar to its counterpart with residual skip connections; i.e. the plain network without floating-point skip connections is just as effective to deploy at inference time. To further promote the gradient flow during back-propagation, we then employ a stochastic structured precision strategy to stochastically sample and quantize sub-networks while keeping other parts full-precision. We evaluate the proposed method on the image classification task over various quantization approaches and show consistent performance increases.

Contents

*Corresponding author. E-mail: `chunhua.shen@adelaide.edu.au`

1 Introduction

Deep neural networks have made great strides in many computer vision tasks such as image classification [11, 19], segmentation [10] and detection [29]. Even though deep and/or wide models can achieve promising accuracy, their huge computational complexity makes them incompatible with energy constrained devices which usually have limited memory bandwidth and computational power. This has motivated the community to design energy-efficient models, often based on quantized precision, aiming not to sacrifice accuracy relative to the full-precision models. In this paper, we propose to tackle two challenges in training accurate low-precision networks.

The first challenge is the non-differentiability of the discrete quantizer. As a result, we cannot directly optimize the discretised network with stochastic gradient descent. The most basic solution is to employ the straight-through estimator (STE) [2], however some recent literature has proposed to relax the discrete quantizer itself to continuous for gradient-based optimization [1, 23]. Even though the discontinuity of the discretization operation during training can be partly solved by smoothing it appropriately, some important information may still be destroyed and lead to an undesirable drop in accuracy. To solve this problem, we propose to design efficient training strategies which are complementary to these quantizer smoothing approaches.

The second challenge is the floating-point storage and operations imported by skip connections in residual type network architectures, which are widely used in existing quantization literature [6, 22, 39, 43, 44]. Because of the floating-point operations, the hardware must preserve floating-point memory, adders and multipliers in advance, which makes the quantized model less hardware-friendly. Moreover, some literature [3, 22, 25] has observed quantizing the skip connections to low-precision will cause apparent performance drop. Instead, we propose to directly train a VGG-style plain network without floating-point operations which brings great benefits to the hardware implementation while still preserving the performance.

We propose the following strategy to simultaneously tackle the above challenges. Our method is to learn a full-precision auxiliary gradient module to provide direct hierarchical supervisions to the low-precision model (see Fig. 1). During training, the original low-precision network is expanded with the auxiliary gradient module to formulate a mixed-precision auxiliary residual network. The auxiliary residual network and the quantized model are jointly optimized with shared convolutional layers and independent batch normalization. Because of the weight sharing strategy, the auxiliary gradients can rectify the approximated gradients direction due to the discontinuity of the quantizer. During testing, only the original low-precision network is utilized for inference. Interestingly, the proposed auxiliary gradient module has two important effects. First of all, the learned auxiliary gradients can compensate the information loss from the discrete quantizer and promote the convergence for general fixed-point approaches. Moreover, it can be used to remove the full-precision skip connections in low-precision models, which results in hardware-friendly quantized models.

To further improve low-precision network training, we upgrade the proposed auxiliary gradient training strategy with a stochastic structured precision scheme which randomly select a portion of the model (*i.e.*, layers, blocks) and activations or weights to quantize while keeping other parts full-precision. We find that this scheme is complementary to the auxiliary gradient training and combining both leads to further performance improvement.

In summary, our contributions are summarized as follows:

- We propose an auxiliary gradient module which has a "two birds, one stone" effect. On one hand, it provides direct gradients to promote the convergence of the original quantized model. On the other hand, it can assist to remove the full-precision skip connections to obtain a completely quantized low-precision model, which brings great benefits to hardware devices.
- We employ a stochastic structured precision scheme to further improve the auxiliary gradient training.
- Extensive experiments show the effectiveness and robustness of the proposed training approaches on the image classification task over various existing quantization approaches.

2 Related Work

Network quantization. Quantization can be categorized into fixed-point quantization and binary neural networks (BNNs), in which fixed-point quantization can be divided into uniform and non-uniform. Uniform approaches [41,43] design quantizers with a constant quantization step. To reduce the quantization error, non-uniform strategies [4, 16, 39] propose to learn the quantization intervals by jointly optimizing parameters and quantizers. A fundamental problem of quantization is to approximate gradients of the non-differentiable quantizer. To solve this problem, some works have studied relaxed quantization [1, 23, 36, 43]. Most recently, some recent literature employs reinforcement learning to search for the optimal bitwidth for each layer [5, 35, 37]. BNNs [14, 28] constrain both weights and activations to binary values (*i.e.*, +1 or -1), which brings great benefits to specialized hardware devices. The development of BNNs can be classified into two categories: (i) a focus on improving the training of BNNs [12, 22, 28, 33]; (ii) multiple binarizations to approximate the full-precision tensor or structure [9, 20, 21, 33, 44]. In this paper, we propose general training approaches that work on all categories of quantization approaches.

Weight sharing. Weight sharing has been attracting increasing attention for efficient, yet accurate computation. In visual recognition, region proposal networks (RPN) in Faster-RCNN [29] and Mask-RCNN [10] share the same backbone with task-specific networks, which greatly saves testing time. For neural architecture search, ENAS [26] allows parameters to be shared among all architectures in the search space, which saves orders of magnitude GPU hours. In the network compression field, weight/activation quantization intends to partition the weight/activation distribution into clusters and use the centers of clusters as the possible discrete values. This strategy can be interpreted as a special case of weight sharing. Different from these approaches, we propose to utilize weight sharing for jointly optimizing the full-precision auxiliary gradient module and the original low-precision network to improve the accuracy of the latter quantized model.

Dropout strategies. Dropout [32], Maxout [8], DropConnect [34] and DropIn [31] are a category of approaches that propose to stochastically drop intermediate nodes or connections during training to prevent the network from overfitting. Huang [13] further propose stochastic depth regularization via randomly dropping a subset of layers during training. And Dong [7] propose to randomly quantize a portion of weights to low-precision in the incremental training framework [40]. The method in [7] 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, Zhuang *et al.* [43] 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 training sacrifices the efficiency greatly. In contrast, we improve the progressive quantization into only a single stage. Our study shows that this extended scheme is complementary to the proposed auxiliary gradient training.

3 Method

We now describe the proposed strategy: in Sec. 3.1, we explain the auxiliary gradient module design and optimization, while in Sec. 3.2, we describe the stochastic structured precision for relaxed optimization.

3.1 Auxiliary gradient module

The module is designed to solve two challenges simultaneously. First, the quantizer $q(\cdot)$ (*e.g.*, $\mathrm{rounding}(\cdot)$, $\mathrm{sign}(\cdot)$) receives the continuous signal as input and outputs discrete values. This process is non-differentiable and the gradients can only be approximated, so we intend to learn additional signals to compensate the information loss. Second, we want to obtain a quantized network with entire fixed-point operations by removing the floating-point skip connections. To solve the above two problems, we propose an auxiliary gradient module that simulates the effects of the skip connections.

3.1.1 Module design

The commonly used layer-ordering for a quantized network [4, 28, 39] is BN$\rightarrow$ ReLU $\rightarrow q(\cdot) \rightarrow$ QConv, where QConv denotes fixed-point convolution and this pre-BN building block is shown in Fig. 1 (a). We also illustrate the original quantized network that consists of these building blocks in Fig. 1 (c). To provide direct gradients that can compensate the information loss of the quantized convolutional layers in the low-precision network, we propose to expand the low-precision network with the full-precision auxiliary gradient module during training. As a result, the whole training framework can now be formulated into two networks: the original low-precision network F and its expanded mixed-precision version H with auxiliary gradient module incorporated, where H shares the parameters of F (see Fig. 1 (b)).

In particular, we add P additional branches on top of the output feature maps $\{\mathbf{O}_p\}_{p=1}^{P}$ of the corresponding convolutional layers in F, respectively. Let $\{l_1, ..., l_P\}$ be the layer indexes where we add the auxiliary gradient branches. For the p-th branch of the gradient module, it receives the feature map $\mathbf{O}_p$ of the l_p-th layer from F and outputs an adapted feature representation $\phi_p(\mathbf{O}_p)$, where $\phi(\cdot)$ is a trainable adaptor. The adaptor could be comprised of one or a few full-precision convolutional layers, we adopt a simple 1×1 convolutional layer followed by a batch normalization layer in this paper.

In the mixed-precision network H, the outputs of the adaptor at different branches are sequentially aggregated in the same fashion as in the residual network. Formally, let g_p denotes the aggregated feature up to the p-th branch. It is calculated by adding the adapted feature $\phi_p(\mathbf{O}_p)$ from F and the $(p-1)$-th aggregated feature in H:

$$g_p = \phi_p(\mathbf{O}_p) + g_{p-1}. \tag{1}$$

For clarity of notation, we omit the initial pre-processing and final classification steps.

There are two situations to discuss: 1) When F has skip connections, the gradient module in H provides more skip connections for F, which has been proven to be effective for training low-precision networks [3, 22, 44]. Moreover, the direct full-precision auxiliary gradients can help rectify the approximate gradients updating direction of the low-precision model; 2) When F is without skip connections, H can be treated as a residual network with a low-precision path F. In other words, we intend to utilize a residual network to implicitly increase the gradient paths of a plain network via weight sharing to facilitate its optimization. The auxiliary residual network H and the plain low-precision network F are jointly optimized to promote the convergence of F. In the testing phase, only the plain network F is used for inference without the auxiliary gradient module, which is illustrated in Fig. 1 (c).

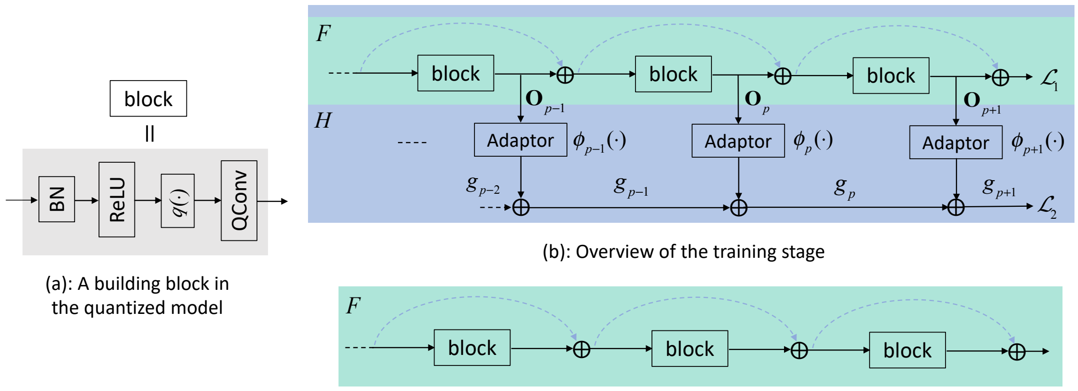
3.1.2 Optimization

To jointly optimize the two networks, we formulate a multi-task objective:

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2, \tag{2}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are objectives for networks F and H, respectively, and λ is a balancing hyperparameter. For the classification task, $\mathcal{L}_1$ and $\mathcal{L}_2$ are both set to the cross-entropy loss.

There is some subtlety to the optimization: in network F, all convolutional layers are quantized while network H consists of a mixture of low-precision and full-precision convolutional layers. However, the BN statistics of the full-precision and low-precision convolutional layers differ a lot due to the quantization process. As a result, the BN layers should not be shared between F and H. To solve this problem (similar to [15, 38]) we assign independent batch normalization parameters for F and H respectively. By



(a): A building block in the quantized model

(b): Overview of the training stage

(c): Overview of the testing stage

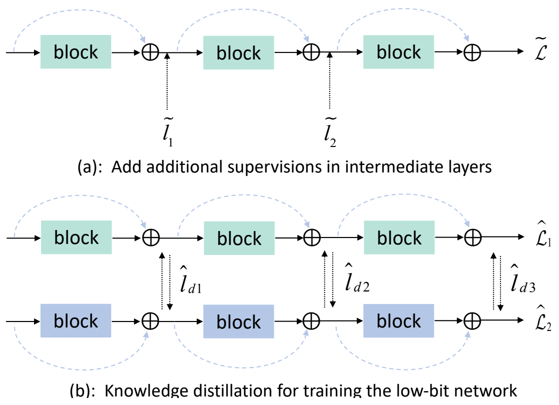
Figure 1: Overview of the proposed framework. Green color represents low-precision operations while blue color denotes full-precision operations. The low-precision network is represented by F and we combine F with the floating-point auxiliary gradient module to formulate an auxiliary residual network H. During the training stage, F and H are jointly optimized with separate batch normalization, where H shares the convolutional layers of F. Only the learnt quantized network F are used during testing. *The floating-point skip connections are denoted by dashed curves since they should be removed when F is a plain network but preserved when F is a residual network.*

privatizing all BN layers, we can resolve the feature distributions inconsistency via independently normalizing the feature mean and variance for the two networks. The extra BN layers of H are only used for training but are all removed during testing, which introduces no extra additional runtime and memory cost for the low-precision network F. Since BN layers are independent in the two networks, we conduct forward calculation for F and H separately in each iteration and accumulate all back-propagated gradients for updating the parameters. As a result, the gradients of the shared convolutional layers are a weighted average from F and H. The optimization process is summarized in Algorithm 1.

Algorithm 1: Joint training approach w.r.t. the low-precision main network F and the mixed-precision auxiliary residual network H.

```
Input: Current mini-batch $\{\mathbf{x}^t, \mathbf{y}^t\}$; convolution and
       BN parameters $\{\mathbf{W}_m^t, \mathbf{P}_m^t\}$ w.r.t. the $p$-th layer
       of the low-precision network $F$; shared
       convolution, adaptor and BN parameters
       $\{\mathbf{W}_m^t, \mathbf{W}_a^t, \mathbf{P}_a^t\}$ w.r.t. the $p$-th branch of the
       auxiliary residual network $H$;
Output: Updated parameters
        $\{\mathbf{W}_m^{t+1}, \mathbf{P}_m^{t+1}, \mathbf{W}_a^{t+1}, \mathbf{P}_a^{t+1}\}$; learning rate
        $\eta^{t+1}$.
1 Obtain the quantized weight $\mathbf{Q}_m^t = q(\mathbf{W}_m^t)$;
2 $\mathbf{y}_m^t = \mathrm{Forward}(\mathbf{x}^t, \mathbf{Q}_m^t, \mathbf{P}_m^t)$;
3 Compute the loss $\mathcal{L}_1(\mathbf{y}^t, \mathbf{y}_m^t)$;
4 $\frac{\partial \mathcal{L}_1}{\partial \mathbf{Q}_m^t}, \frac{\partial \mathcal{L}_1}{\partial \mathbf{P}_m^t} = \mathrm{Backward}(\frac{\partial \mathcal{L}_1}{\partial \mathbf{y}_m^t}, \mathbf{Q}_m^t, \mathbf{P}_m^t)$ ;
5 $\mathbf{y}_a^t = \mathrm{Forward}(\mathbf{x}^t, \mathbf{Q}_m^t, \mathbf{W}_a^t, \mathbf{P}_a^t)$ ;
6 Compute the loss $\mathcal{L}_2(\mathbf{y}^t, \mathbf{y}_a^t)$;
7 $\frac{\partial \mathcal{L}_2}{\partial \mathbf{Q}_m^t}, \frac{\partial \mathcal{L}_2}{\partial \mathbf{W}_a^t}, \frac{\partial \mathcal{L}_2}{\partial \mathbf{P}_a^t} = \mathrm{Backward}(\frac{\partial \mathcal{L}_2}{\partial \mathbf{y}_a^t}, \mathbf{Q}_m^t, \mathbf{W}_a^t, \mathbf{P}_a^t)$;
8 Compute the gradients of $\mathbf{Q}_m^t$:
  $\nabla \mathbf{Q}_m^t = \frac{1}{1+\lambda}(\frac{\partial \mathcal{L}_1}{\partial \mathbf{Q}_m^t} + \lambda \frac{\partial L_2}{\partial \mathbf{Q}_m^t})$
9 Update parameters using Adam;
```

3.1.3 Relationship to other approaches



(a): Add additional supervisions in intermediate layers

(b): Knowledge distillation for training the low-bit network

Figure 2: Overview of the two related approaches. Green color represents low-precision operations while blue color denotes floating-point operations.

Training with additional losses. To provide auxiliary signals for improving convergence, a straightforward way is to add multiple losses to the intermediate layers, which is shown in Fig. 2 (a). Then the final objective becomes:

$$\widetilde{\mathcal{L}}_{obj} = \widetilde{\mathcal{L}} + \sum_{i=1}^{M} \alpha_i \widetilde{l}_i, \tag{3}$$

where $\widetilde{\mathcal{L}}$ is the task loss, α_i and $\widetilde{l}_i$ are the i-th scale and auxiliary loss, respectively. The multiple auxiliary losses serve as regularizers and can provide direct gradients during training. However, they cannot provide hierarchical gradient paths as the auxiliary gradient module. Moreover, the scales as well as the positions of the additional supervisions are very heuristic and may have negative impact to the final performance. In the proposed approach, a single auxiliary objective is added to the network H. In addition, the auxiliary gradient module allows the final supervision directly propagate back to multi-level intermediate layers.

Knowledge distillation. Recently, there are several works that propose to use knowledge distillation (KD) to improve the performance of the low-precision network [24, 27, 43].

In particular, a student low-precision network learns to mimic the knowledge of a full-precision teacher network, where both networks can be mutually updated (see Fig. 2 (b)). The objective can be formulated as

$$\hat{\mathcal{L}}_{obj} = \hat{\mathcal{L}}_1 + \hat{\mathcal{L}}_2 + \sum_{i=1}^{M} \beta_i \hat{l}_{di}, \quad (4)$$

where $\hat{\mathcal{L}}_1$ and $\hat{\mathcal{L}}_2$ are task-specific objectives for student and teacher networks respectively, β_i and $\hat{l}_{di}$ represent the i-th scale and distillation loss. Compared with KD approaches, the proposed auxiliary gradient module has several advantages. First, the extra training overhead is smaller. The main network F shares the weights with H, where the only additional parameters are from the adaptors and separate batch normalization layers. In contrast, KD introduces an additional teacher network to be optimized, which is usually deeper than the student low-precision model, and the final performance can be sensitive to the positions of adding the guided signals. Furthermore, the auxiliary gradient module can mimic the effect of skip connections to optimize a plain network. It enables us to train an energy-efficient completely quantized model with comparable performance with the original residual architecture.

3.2 Stochastic structured precision

Algorithm 2: Stochastic structured precision training algorithm.

```
Input: Training data $\{\mathbf{x}^t, \mathbf{y}^t\}$; low-precision network
       $F$ with parameters $W^t$; stochastic ratio $\delta^t$ and
       decay rate $\mu$.
Output: Updated parameters $W^{t+1}$; stochastic ratio
        $\delta^{t+1}$.
1  Partition $F$ into $N$ fragments $\{f_1, ..., f_N\}$;
2  if $\delta^t > 0$ then
3      Obtain the binary indicator matrix $\mathbf{B}^t$ via uniform
       sampling with probability $\delta^t$;
4      Partition the network $F$ 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
       $\widetilde{Q}^t = \{q(W^t_{qwa}), q(W^t_{qw}), W^t_{qa}, W^t_r\}$ accordingly;
6  else
7      $\widetilde{Q}^t = q(W^t)$;
8  $\widetilde{\mathbf{y}}^t = \mathrm{Forward}(\mathbf{x}^t, \widetilde{Q}^t, \mathbf{B}^t)$;
9  Compute the loss $\mathcal{L}(\mathbf{y}^t, \widetilde{\mathbf{y}}^t)$;
10 $\frac{\partial \mathcal{L}}{\partial \widetilde{Q}^t} = \mathrm{Backward}(\frac{\partial \mathcal{L}}{\partial \widetilde{\mathbf{y}}^t}, \widetilde{Q}^t, \mathbf{B}^t)$;
11 Update parameters $W^{t+1}$ using Adam;
12 $\delta^{t+1} = \delta^t - \mu$;
```

The gradient approximation of the quantization function is noisy and may not be the right updating direction. And most existing quantization approaches quantize all the weights and activations all together in each iteration. Recent studies show that progressively or stochastically quantize a part of the network [7,40], or quantize weights and/or activations [43] leads to better convergence of the low-precision network training.

To further improve the network training with the auxiliary gradient module, this section develop of an extension of our method by employing a stochastic structured precision. The term "stochastic structure" means that we will randomly choose a network structural aspect, i.e., layers, blocks, activations or weights to quantize and keep the rest full precision. The specific scheme is elaborated as follows.

Suppose we decompose the low-precision network F into N fragments $F = \{f_1, ..., f_N\}$, where f_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 $G_q = \{f_{q1}, ..., f_{qN_q}\}$ and a full-precision set $G_r = \{f_{r1}, ..., f_{rN_r}\}$, which satisfies the condition:

$$G_q \cup G_r = F, \text{ and } G_q \cap G_r = \emptyset. \quad (5)$$

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

In our method, we randomly partition F 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 will be quantized and otherwise will be kept as full precision. We linearly decrease δ to 0 to ensure the whole network quantized in the end. Note that this procedure implicitly achieves the effect of [40] but without the need of multi-round training.

To further increase the randomness in quantizing f, we can stochastically choose whether to quantize weights or activations or both of them. This can be implemented by randomly sample a 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. In this way, our method can achieve the effect of progressive training as proposed in [43].

In Sec. 4.5.1, we will explore the effect of different structure choices of f as well as the extent of randomness to the final performance.

4 Experiments

We define several methods for comparison as follows: **AG**: We jointly optimize the auxiliary gradient module and the original low-precision network as described in Sec. 3.1. We test it on two scenarios: low-precision network with or without skip connections. **SSP**: It corresponds to the stochastic structured precision strategy in Sec. 3.2. **AG + SSP**: We further combine the two strategies to form the complete approach. To verify the effectiveness of the proposed training approaches, we experiment on various representative quantization approaches, including uniform fixed-point approach DoReFa-Net [41], non-uniform fixed-point method LQ-Net [39], as well as binary neural network approaches BiReal-Net [22] and GroupNet [44]. We perform experiments on two standard image classification datasets, including CIFAR-100 [18] and ImageNet [30].

4.1 Implementation details

Following previous approaches [14, 39, 41–43], we quantize all the convolutional layers to ultra-low precision except the first and last layers. However, to achieve a completely quantized network, we keep the first convolutional layer and the last fully-connected layer to 16-bit. We first pre-train the full-precision counterpart as initialization and then fine-tune the quantized model. For all ImageNet experiments, training and testing images are resized to 256×256, and 224×224 patches are randomly cropped from an image or its horizontal flip, with the per-pixel mean subtracted. We use the single-crop setting for testing. No bias terms are used. We use SGD optimizer for the pre-training stage. For the fine-tuning stage, we adopt the Adam optimizer [17]. The mini-batch size is set to 256. We train a maximum 35 epochs and decay the learning rate by 10 at the 25-th and 30-th epochs. For training the fixed-point methods [39, 41], the learning rate is initialized by 1e-3. For BNNs [22, 44], the initial learning rate is set to 5e-4. For the *AG* based experiments, we set λ to 1.0. In practice, we add one auxiliary branch at the end of each block (*e.g.*, a residual block in [11]). For the *SSP* based experiments, the stochastic ratio δ is initialized to 0.5 and linearly decayed to 0 at the 20-th epoch. Our implementation is based on Pytorch.

4.2 Effect of the auxiliary gradient module

To investigate the effectiveness of the proposed auxiliary gradient module, we perform experiments on two settings: the low-precision model with or without floating-point skip connections.

With skip connections. We first elaborate the cases with skip connections. The results are reported in Table 1. By combining the baseline (with skip connections) with *AG*, we can observe steadily performance increase compared with the original baseline. This strongly supports that the learned floating-point auxiliary gradients can facilitate the convergence of the low-precision model. The gradients of the shared weights are averaged from both the gradient auxiliary module and the original low-precision network to achieve more accurate updating direction. Moreover, increasing gradient paths are important to solve the non-differentiability of the discrete quantization process especially for binary neural networks [3, 22]. Note that we break the records of BNNs by further improving the current state-of-the-art GroupNet. To make it more clear, we plot the convergence curves in Fig. 3 (a). From the figure, we can observe that *AG* can provide a good initialization for finetuning and assist the convergence.

Without skip connections. We then analyze training a plain low-precision network without skip connections. The results can be seen in Table 2. *plain* represents we directly optimize a low-precision plain network without skip connections. By comparing *plain* and *plain + AG*, we observe apparent accuracy increase by incorporating *AG*. For example, in ResNet-34 based experiments, introducing *AG* can boost the accuracy by $\sim$5%. However, we still find performance gap between the full-precision accuracy. This can be attributed to two assumptions of skip connections. First, the skip connections may improve the convergence of training, and this can be proven by the performance improvement by incorporating the additional *AG*. Second, the skip connection and the activations after one convolution are added through a tensor addition. Then the representational capability (*i.e.*, the value range) of each entry in the added activations is significantly enhanced. In other words, the plain network has less representability than its residual counterpart. However, the performance gap is within 1% which is still acceptable but will bring great benefits for hardware implementations. We further plot the convergence curves in Fig. 3 (b). From the figure, we can observe that *AG* can steadily improve the plain network baseline in all epochs.

Table 1: Accuracy (%) of different comparing methods on the ImageNet validation set. All the cases keep skip connections during testing.

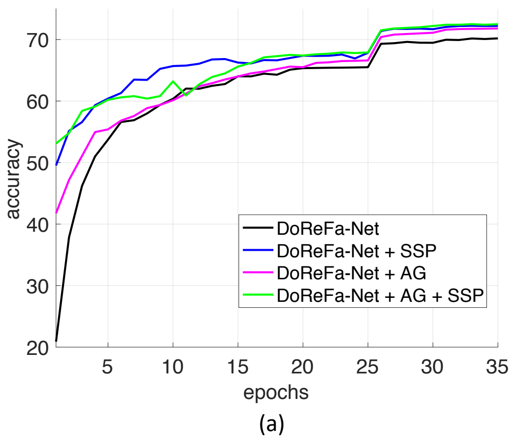
model	method	top-1 acc.	top-5 acc.
ResNet-50	DoReFa-Net (2-bit)	70.2	89.1
	DoReFa-Net + AG	71.8	90.6
	DoReFa-Net + SSP	72.2	90.8
	DoReFa-Net + AG + SSP	**72.5**	**90.9**
ResNet-50	LQ-Net (3-bit)	74.2	91.6
	LQ-Net + AG	75.4	92.4
	LQ-Net + SSP	75.1	92.3
	LQ-Net + AG + SSP	**75.6**	**92.6**
ResNet-18	BiReal-Net	56.4	79.5
	BiReal-Net + AG	58.6	81.2
	BiReal-Net + SSP	58.8	81.2
	BiReal-Net + AG + SSP	**58.9**	**81.4**
ResNet-18	GroupNet (5 bases)	64.8	85.7
	GroupNet + AG	66.0	86.5
	GroupNet + SSP	65.9	86.3
	GroupNet + AG + SSP	**66.2**	**86.8**

Table 2: Accuracy (%) of the proposed approaches on the ImageNet validation set. All the cases are 2-bit and without skip connections except for the baselines. We can observe that the auxiliary gradient module can significantly improve the plain network performance.

model	method	top-1 acc.	top-5 acc.
DoReFa-Net on ResNet-18	baseline (2-bit)	62.7	84.2
	plain	59.5	82.1
	plain + AG	61.8	83.7
	plain + AG + SSP	**62.2**	**84.0**
DoReFa-Net on ResNet-34	baseline (2-bit)	66.2	86.6
	plain	60.4	82.5
	plain + AG	64.9	85.8
	plain + AG + SSP	**65.5**	**86.2**
LQ-Nets on ResNet-34	baseline (2-bit)	69.8	89.1
	plain	63.5	84.6
	plain + AG	68.6	88.5
	plain + AG + SSP	**69.3**	**88.8**

4.3 Effect of the structured stochastic precision

Here we further explore the effect of the structured 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



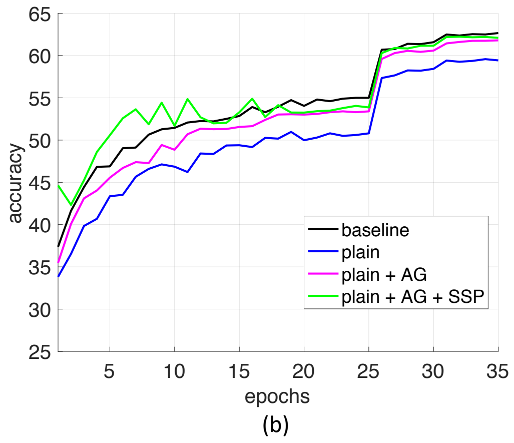


Figure 3: (a): Accuracy(%) w.r.t. 2-bit DoReFa-Net with ResNet-50 on ImageNet. The quantized model has skip connections. (b): Accuracy(%) w.r.t. 2-bit DoReFa-Net with ResNet-18 on ImageNet. The quantized model is without skip connections. The learning rate is decayed by 10 at the 25-th and 30-th epochs.

cases unless special explanations. The results are reported in Table 1. By combining the baseline methods with *SSP*, 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 progressive quantization [43] to relax the discrete quantization function effectively. Moreover, the proposed progressive strategy only requires one training stage without fine-tuning the model in many training rounds.

4.4 Effect of combining AG and SSP

We further combine *AG* and *SSP* and report the full performance in Table 1 and Table 2. From the results, we can conclude that *SSP* is complementary to *AG* which can further benefit the performance of *AG*. For example, for 2-bit DoReFa-Net with ResNet-18 on ImageNet, *AG* + *SSP* can reduce the gap between a quantized plain network and its residual counterpart to 0.5%. Moreover, we illustrate the convergence curves in Fig. 3. *plain* + *AG* + *SSP* fluctuates at the beginning and gradually becomes stable due to the linearly decayed randomness. And its accuracy is consistently better than *plain* and *plain* + *AG*.

4.5 Ablation study

4.5.1 Comparison with related approaches

In this section, we compare the auxiliary gradient module with the related approaches discussed in Sec. 3.1.3. *additional losses* and *KD* correspond to the "Training with additional losses" and "Knowledge distillation" strategies, respectively. We experiment based on the 2-bit DoReFa-Net with ResNet-18 and ResNet-34 on ImageNet. The results are reported in Table 3. For training *additional losses*, we evenly add the cross-entropy losses at the end of residual blocks which include downsampling and set the balancing parameter α to 0.01. For training *KD*, we follow the same setting in [43] and the teacher network is set to be the same as the student network.

When the low-precision model is a plain network, we can see that *plain* + *AG* achieves the best performance. For example, *plain* + *AG* outperforms *plain* + *KD* by 3% on ResNet-34. It can be attributed to the design of the auxiliary gradient module which mimics the structure of skip connections during training. When it comes to the case where the fixed-point model is with full-precision skip connections, we can observe that *AG* and *KD* achieve comparable performance while both perform much better than the *additional losses*. And we empirically find that *additional losses* is sensitive to the scales and positions of additional supervisions and have limited contribution to the final performance. For *KD*, both the full-precision teacher network and the low-precision student network are jointly optimized which increases the training burden a lot. In contrast, we only introduce several additional 1×1 convolutional layers and batch normalization layers, where the extra training cost is much smaller. Moreover, the proposed *AG* requires only one hyperparameter and have advantages over *KD* on training low-precision plain networks.

Table 3: Accuracy (%) of different supervision strategies on the ImageNet validation set based on 2-bit DoReFa-Net on ResNet-18 and ResNet-34.

model	method	top-1 acc.	top-5 acc.
ResNet-18	DoReFa-Net (2-bit)	62.7	84.2
	plain + additional losses	59.7	82.1
	plain + KD	60.6	83.0
	plain + AG	**61.8**	**83.7**
	DoReFa-Net + additional losses	62.9	84.1
	DoReFa-Net + KD	64.1	**85.5**
	DoReFa-Net + AG	**64.3**	85.4
ResNet-34	DoReFa-Net (2-bit)	66.2	86.6
	plain + KD	61.9	83.7
	plain + AG	**64.9**	**85.8**

4.5.2 Effect of different SSP 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 4. We explore two different structures to 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. 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 4: 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

4.5.3 Effect of the number of auxiliary gradient paths

In this section, we explore the effect of the number of auxiliary gradient paths to the final performance. We work on 2-bit DoReFa-Net based on ResNet-34 (without skip connections) on ImageNet and the results are shown in Table 5. *blockwise AG* denotes we add an auxiliary branch at the end of each residual block while *layerwise AG* indicates we add one additional path after each convolutional layer. From the results, we empirically find that these two variants achieve similar accuracy. It can be attributed to the blockwise auxiliary gradient module is good enough for simulating the skip connections. Adding more gradient paths can provide more hierarchical supervisions but the improvement is limited.

Table 5: Accuracy (%) w.r.t. different number of auxiliary branches on the ImageNet validation set.

model	method	top-1 acc.	top-5 acc.
ResNet-34	plain + blockwise AG	64.9	85.8
	plain + layerwise AG	**65.1**	**85.9**

4.6 Experiments on CIFAR-100

Finally, we test the proposed approach on smaller scale dataset CIFAR-100 to justify its robustness. The experiment is based on 2-bit DoReFa-Net using ResNet-18. We report the results in Table 6. From the results, we can find that the proposed *AG* can significantly improve the quantized plain model results. Interestingly, *plain* + *AG* can even outperform the baseline that with skip connections in terms of Top-1 accuracy. In other words, we can train a low-precision plain network without skip connections (*i.e.*, completely quantized network) that can compete with the original low-precision residual network counterpart. Moreover, training with *AG* and *SSP* together can further improve the accuracy compared with employing *AG* only, which justifies *SSP* is complementary to the optimization of *AG*.

Table 6: Accuracy (%) of 2-bit DoReFa-Net using ResNet-18 on the CIFAR-100 dataset.

model	method	top-1 acc.	top-5 acc.
ResNet-18	full-precision	70.7	91.3
	baseline (2-bit)	67.6	90.2
	baseline + AG	68.3	90.0
	baseline + AG + SSP	**68.7**	**90.5**
	plain (2-bit)	64.6	88.3
	plain + AG	67.9	90.0
	plain + AG + SSP	**68.4**	**90.3**

5 Conclusion

In this paper, we have proposed an auxiliary gradient module for training low-bitwise convolutional neural networks. In specific, we have proposed to expand the original low-precision network with a full-precision auxiliary gradient module during training. The auxiliary mixed-precision residual network and the original quantized network are jointly optimized with weight sharing and separate batch normalization. This strategy has two effects. On one hand, when low-bitwise model is with skip connections, it can increase the gradient paths to promote convergence. On the other hand, if the low-bitwise model is a plain network, the auxiliary gradient module can mimic the effect of skip connections to aid the training. As a result, we can train a completely quantized network, which brings great benefits to the hardware devices. Moreover, we have also employed a stochastic structured precision strategy to further improve the auxiliary gradient training. We have conducted extensive experiments on various quantization approaches and observed consistent performance increase on the image classification task.

References

[1] Y. Bai, Y.-X. Wang, and E. Liberty. Proxquant: Quantized neural networks via proximal operators. In *Proc. Int. Conf. Learn. Repren.*, 2019. 2

[2] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2

[3] J. Bethge, M. Bornstein, A. Loy, H. Yang, and C. Meinel. Training competitive binary neural networks from scratch. *arXiv preprint arXiv:1812.01965*, 2018. 2, 3, 6

[4] 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.*, pages 5918–5926, 2017. 2, 3

[5] Y. Chen, G. Meng, Q. Zhang, X. Zhang, L. Song, S. Xiang, and C. Pan. Joint neural architecture search and quantization. *arXiv preprint arXiv:1811.09426*, 2018. 2

[6] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2

[7] 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, 5

[8] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proc. Int. Conf. Mach. Learn.*, 2013. 3

[9] 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.*, pages 5955–5963, 2017. 2

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2980–2988, 2017. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. 2, 6

[12] L. Hou, Q. Yao, and J. T. Kwok. Loss-aware binarization of deep networks. In *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[13] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Proc. Eur. Conf. Comp. Vis.*, pages 646–661, 2016. 3

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 4107–4115, 2016. 2, 6

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*, pages 448–456, 2015. 3

[16] 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

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Repren.*, 2015. 6

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 6

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1097–1105, 2012. 2

[20] 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.*, pages 2584–2592, 2017. 2

[21] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 344–352, 2017. 2

[22] 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, 3, 5, 6

[23] 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

[24] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *Proc. Int. Conf. Learn. Repren.*, 2018. 4

[25] E. Park, D. Kim, S. Yoo, and P. Vajda. Precision highway for ultra low-precision quantization. *arXiv preprint arXiv:1812.09818*, 2018. 2

[26] 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

[27] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. In *Proc. Int. Conf. Learn. Repren.*, 2018. 4

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 525–542, 2016. 2, 3

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 91–99, 2015. 2

[30] 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.*, 115(3):211–252, 2015. 6

[31] 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.*, pages 4763–4771, 2016. 3

[32] 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.*, 15(1):1929–1958, 2014. 3

[33] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *Proc. AAAI Conf. on Arti. Intel.*, pages 2625–2631, 2017. 2

[34] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proc. Int. Conf. Mach. Learn.*, pages 1058–1066, 2013. 3

[35] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. Haq: Hardware-aware automated quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 2

[36] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, J. Cheng, et al. Two-step quantization for low-bit neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4376–4384, 2018. 2

[37] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018. 2

[38] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. In *Proc. Int. Conf. Learn. Repren.*, 2019. 3

[39] 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, 3, 5, 6

[40] 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. Repren.*, 2017. 3, 5

[41] 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. 2, 5, 6

[42] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *Proc. Int. Conf. Learn. Repren.*, 2017. 6

[43] 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, 4, 5, 6, 7

[44] 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, 3, 5, 6