Proceedings of Machine Learning Research 157, 2021 ACML 2021

DPOQ: Dynamic Precision Onion Quantization

Bowen Li LIBW@ZJU.EDU.CN
Kai Huang HUANGK@VLSI.ZJU.EDU.CN
Siang Chen CHENSIANG@ZJU.EDU.CN
Dongliang Xiong XIONGDL@ZJU.EDU.CN
Institute of VLSI Design, Zhejiang University, Hangzhou, China
Luc Claesen LUC.CLAESEN@UHASSELT.BE
Engineering Technology - Electronics-ICT Dept, Hasselt University, 3590 Diepenbeek, Belgium

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

With the development of deployment platforms and application scenarios for deep neural networks, traditional fixed network architectures cannot meet the requirements. Meanwhile the dynamic network inference becomes a new research trend. Many slimmable and scalable networks have been proposed to satisfy different resource constraints (e.g., storage, latency and energy). And a single network may support versatile architectural configurations including: depth, width, kernel size, and resolution. In this paper, we propose a novel network architecture reuse strategy enabling dynamic precision in parameters. Since our low-precision networks are wrapped in the high-precision networks like an onion, we name it dynamic precision onion quantization (DPOQ). We train the network by using the joint loss with scaled gradients. To further improve the performance and make different precision network compatible with each other, we propose the precision shift batch normalization (PSBN). And we also propose a scalable input-specific inference mechanism based on this architecture and make the network more adaptable. Experiments on the CIFAR and ImageNet dataset have shown that our DPOQ achieves not only better flexibility but also higher accuracy than the individual quantization.

Keywords: Dynamic Quantization; Neural Network

1. Introduction

Deep neural networks (DNNs) have been proved to be powerful on many artificial intelligence (AI) tasks such as image classification He et al., autonomous driving Grigorescu et al. (2020), natural language processing Goldberg (2016), and so on. DNNs are deployed in a vast amount of hardware platforms, ranging from cloud devices to mobile and even IoT (Internet of Things) devices. With the explosive increasing of platform diversification, deployment problems on how to support varied devices with different competence constraints begin to emerge. Moreover, even running on the same device, there are still many changeable application and hardware scenarios that make the situation more complicated. For example, mobile AI requires to consume less power in low battery levels, and the visual recognition system embedded in autonomous vehicles placing a heavier demand on latency and accurate prediction.

Many neural architecture search (NAS) methods Zoph and Le; Zoph et al.; Real et al. are proposed to assist manual design. They integrate the hardware constraints into search objectives and find candidate networks. However, the optimal network architecture varies significantly under different hardware resources, and it is highly expensive to design specialized DNNs for every scenario. As a result, both the human-based design and NAS are unable to keep up with the demand. To tackle this problem, the networks are required to be configurable and adaptive on account of different hardware devices and dynamic deployment environments. Many dynamic architectures and inference methods have been proposed to make the network more flexible and scalable. Yu et al. trains the slimmable neural networks with different layer widths (number of active channels), and adjust the amount of channels during inference to meet resource budgets.

Inspired by slimmable neural networks, many works Cai et al. focus on exploring the adaptation of other dimensions including: depth, width, kernel size, and resolution. Among them, dynamic precision has not been well studied in the previous work. AdaBits Jin et al. simultaneously train models under different bit-widths with shared full-precision weights. After training, the precision of the network's representations can be adjusted on the fly. However, the bit-width of the network is "static" for all inputs, no matter the sample is easy or hard. And they adjust the model by quantizing the shared weights to the expected bit-width, which introduces an alignment problem when switching between two quantization bit-widths. They solve it by using the floor rounding function, but it damages the accuracy and cause non-convergence in low bit-width. Zhang et al. (c) propose precision gating (PG), a method enabling dual-precision execution of DNNs. They split the features into two precision, only important features are assigned with higher precision classes. The criteria of importance is decided by whether the feature value is greater than a learnable gating threshold.

In this paper, we propose the dynamic precision onion quantization (DPOQ) as illustrated in Figure 1. The low-precision network is wrapped in the high-precision network and the bit-width is increased layer by layer like an onion. Our DPOQ reuses low-precision intermediate results when doing high-precision running. For example, both 2-bit and 4-bit results can be obtained with 2-times 2bit forward running. Different from AdaBits, the parameters in our DPOQ are already quantized, and we adjust the precision by cutting or adding part of the network. The key differences between PG and our DPOQ are that our DPOQ applies multi-precision to the weights and outputs the results of all precision networks. We investigate and examine separate and joint training for the DPOQ. And we discover that both approaches are single-faceted and only perform well in part of precision. We tackle this by using a compromised update rule with scaled gradients. For better data interaction between different precision, we propose a simple and effective approach named precision shift batch normalization (PSBN). We utilize the affine transformation in the original Batch Normalization layer Ioffe and Szegedy (2015) and add an extra output for precision shifted data. Moreover, the low-precision outputs can be treated as an early exit. If the low-precision results are not satisfactory for some images, the high-precision network can reuse the intermediate results from the low-precision network, and only a difference bit network computation is needed to get the high-precision results. Based on that, we take a new variable confidence into consideration and put forward a dynamic precision inference mechanism. Our contributions are summarized as follows:

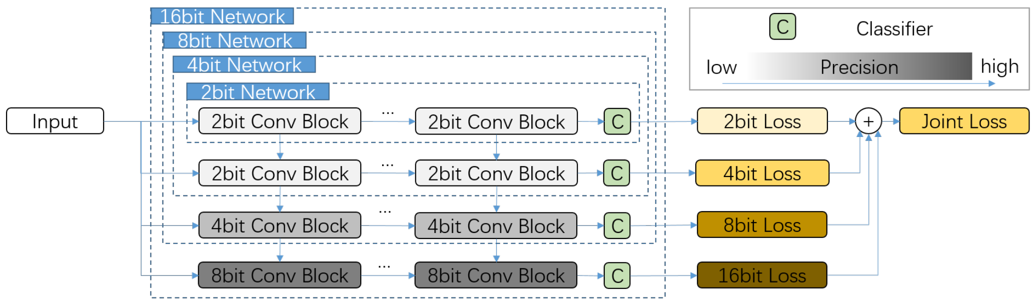


Figure 1: The illustration of our DPOQ network of four precision: 2,4,8 and 16 bits

- We design the dynamic precision onion quantization (DPOQ), which can adjust its weight precision on the fly according to hardware constraints or accuracy demands.
- We propose precision shift batch normalization (PSBN) to alleviate the distribution difference among different precision networks.
- We explore an early exit based scalable inference method, which considers the difficulty of different inputs and makes the DPOQ network inference more cost-effective.

2. Related Work

2.1. Neural Network Quantization

To achieve better performance and more accurate prediction, the networks grow wider and deeper, and the network compression is an essential step for deployment. Quantization is one of the most widely adopted compression methods. It reduces the model size and speeds up the inference by the means of converting floating-point arithmetic of neural networks into fixed-point. Quantization methods can roughly be divided into post-training quantization Jacob et al.; Banner et al.; Choukroun et al.; Nagel et al. and quantization-aware quantization Hubara et al.; Courbariaux et al.; Rastegari et al.; Zhou et al. (2016) , according to the order of quantization and training. BNNs Hubara et al. binarize weights and activations into {-1,+1}, and the energy-consuming multiplication-accumulations can be replaced by lightweight *xnor* and *popcount* operations. DoReFa Zhou et al. (2016) quantize the weights, activations and gradients into low bit-width. Jacob et al. propose an integer-arithmetic-only quantization that limits both weights and activation to 8-bit.

Quantization-aware training methods make the model to adapt to the quantization noise through training, but normally only support a specific precision at a time. Post-training quantization methods need no further re-training, but suffer from significant performance degradation for low bit-width quantization. Developing a single model applicable for different bit-widths is still an open problem that has not been well-examined.

2.2. Dynamic Inference

Dynamic inference is a technique to adjust the network structure at runtime to satisfy different resources and tasks demands. We subdivide the concept of dynamic inference into

input-dependent and input-independent according to the fact if the network architecture is relevant to the input or not.
Input-independent. The network architecture is mainly determined by the hardware resource constraints. The network is trained to support versatile architectural configurations, and it can be specialized in different deployment scenarios in need. For example, the different scenarios here can be concretized into different constraints, for instant: memory, runtime latency and power consumption, and we can select part of the network for inference without re-training. Slimmable neural networks Yu et al.; Yu and Huang (a,b) train the network executable with different widths, and allow instant and adaptive accuracy-efficiency trade-offs at runtime. Cai et al. propose once-for-all, which decouples training and search to support diverse architectural settings including depth, width, kernel size, and resolution. AdaBits Jin et al. enables adaptive bit-widths of weights and activations by using a joint quantization approach and the switchable clipping level.
Input-dependent. Even in the same dataset, it has different difficulty levels to classify different inputs. Running "easy" samples with intensive computational intensive models may cause the network "overthinking" Kaya et al., which is not only wasteful but also harmful. Based on that, the input-dependent methods are more flexible and can adjust the network architecture depending on the input. They mitigate the "overthinking" problem by adding auxiliary classifiers to the internal layers of a standard network and forcing the majority of input patterns to exit at the branches based on the confidence. The multiple "early exits" method Zhang et al. (b); Teerapittayanon et al. not only cuts the average inference cost but also brings some accuracy improvements. SkipNet Wang et al. learns to dynamically skip redundant layers on a per-input basis. Challenging images are executed with more layers than easy images. MSDNets Huang et al. and RANetsYang et al. use multi-scale dense structure for efficient inference. They compress the input samples to multi-scales, and an easy sample can be classified rapidly from a low-resolution feature map.

3. Dynamic Onion Quantization

The output of a convolution layer can be formulated as:

$$O = I * W + B. \tag{1}$$

Where I, W and B are the inputs, weights and bias respectively, and we omit the bias in the remainder for simplicity. The weight W can be quantized to two different bit-widths hb and lb ($hb > lb$), and the fixed-point formats after quantization are W_{hb} and W_{lb}. Suppose the quantization is linear and the W_{lb} is identical to the most-significant lb bits of W_{hb}. We define the least-significant $hb - lb$ bit of W_{hb} as:

$$W_{hb-lb} = W_{hb} - W_{lb} * 2^{(hb-lb)}. \tag{2}$$

Then we can reformulate the high-precision output O_{hb} into two lower-precision calculations, and reuse the low-precision convolution results to reduce computations:

$$\begin{aligned} O_{hb} &= I * W_{hb} \\ &= I * (W_{lb} * 2^{(hb-lb)} + W_{hb-lb}) \\ &= O_{lb} + O_{hb-lb}. \end{aligned} \tag{3}$$

However, due to the nonlinear function $F(.)$ in the activation layer, the output after the first convolution layer $O^n_{hb} = F(O^{n-1}_{hb}) * W^n_{hb}, (n > 1)$ can not be directly reformulated with O^n_{lb} and O^n_{hb-lb}. For architectural consistency, we make an approximation to the expression of the high-precision output, and the network will learn appropriate parameters during training:

$$O^n_{hb} \approx O^n_{lb} + O^n_{hb-lb}. \tag{4}$$

In most networks, a basic block is composed of a convolution layer, a batch-normalization (BN) layer, and a ReLU layer. Based on the above approximation, we propose our dual-precision basic block architecture in Figure 2(a)subfigure. O_{lb} and O_{hb-lb} are the outputs of $Conv_{lb}$ and $Conv_{hb-lb}$ separately, and the two outputs are added to get the O_{hb} before sending it to the next layer. Since we divide the low-precision results and high-precision results into two different data-paths and assign each precision with an individual BN layer, we do not have the distribution consistency problem as slimmable neural networks have.

Furthermore, the dual-precision network can be easily extended to a multi-precision network, and each new precision only calculates the difference precision from the previous one. With the basic block, we build up a 4-precision dynamic precision onion quantization (DPOQ) network as illustrated in Figure 1. The total number of weight bits is 16 and it consists of 4 parts: 2 bits, 2 bits, 4 bits, and 8 bits. On the basis of our reuse strategy, the precisions of the four sub-network outputs are 2 bits, 4 bits, 8 bits, and 16 bits. The architecture has two directions of data-path: the horizontal direction and the vertical direction. The horizontal direction data-path denotes that the network goes deeper. The input samples are sent to each bit-mode network, and each sub-network has a classifier outputting its predicted results. The vertical direction data-path denotes the data reuse. The high-precision networks reuse the intermediate convolution results from lower-precision networks.

This structure gives three advantages. First, the precision of the DPOQ network is easy to be adjusted according to hardware and resource constraints. We optimize all precision network loss functions during training, and the parameters in the DPOQ network have been already quantized. As a result, the precision switching can be achieved by simply cutting or adding part of the network in deployment. And this architecture is also extensible and reusable. When a new precision is added to a pre-trained DPOQ network, we can only train the new part and fix the rest part of the network. Second, the low-precision outputs can be seen as an early exit of high-precision ones. The network will first predict a sample with the lowest precision sub-network. If the lowest precision output is confident enough, the sample can exit from the network. It can avoid the high computational cost induced by performing convolutions on a higher precision network and bring significant reductions of the inference cost. Then if the low-precision output is unsatisfactory or unreliable, it is easy to calculate the higher precision outputs based on low-precision intermediate convolution results. Third, we can calculate the ensemble prediction results with almost no extra cost. The parameters and operations of the BN and ReLU layer are negligible comparing to the convolution layer. We can get both high-precision and low-precision results at the cost of high-precision running.

3.1. Training Strategy

The training of our architecture is similar but nonidentical to the training of multi-exits networks. In short, the multi-exits network is composed of a backbone network and several branch classifiers. And customarily, the branch classifiers are not used alone, and if the early classifier is unable to output a confident result, later classifiers will make up for that. In some works Szegedy et al., the early-exits are only used to improve the performance of the backbone network during training, because they help to reduce the tendency to over-fitting and vanishing gradients. Therefore, more attention is paid to the accuracy of the final backbone network output than of those auxiliary classifiers. Instead, the PSBN do not have a distinction between backbone and branch networks, all classifiers of our network are required to play the pivotal role for different resource-constrained conditions.

There are two approaches to train the multi-exits network: joint training and separate training. The main difference between the two approaches is whether the gradient of the branch outputs loss is transmitted to the backbone network in backpropagation. The separate training is applicable to the situations in which the backbone network is provided beforehand. And the loss function of both approaches can be formulated as a weighted sum of per-output losses:

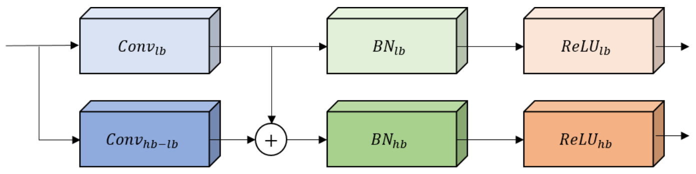
$$loss = \sum_{k \in BitList} \alpha_k CrossEntropy(y_k, y), \tag{5}$$

where α_k weighs the contribution of the kth classifier loss. y_k and y are the outputs of kth classifier and the ground truth separately. Since the different demands mentioned above, multi-exits methods are not directly applicable to our network. On the one hand, the joint training may cause poor performance at low precision. More specifically, from Figure 1 we can see, the 2-bit output is only calculated from a 2-bit network. However, the convolution blocks in the 2-bit network are used in all the 4 precision outputs. Suppose the α_k is all ones for each precision. Following the chain rule, the gradients of 2-bit network parameters θ_2 can be computed as: $\nabla_{\theta_2} L = \partial(L_2 + L_4 + L_8 + L_{16})/\partial\theta_2$. It has three additional terms coming from higher precision loss, which means only about a quarter of the total gradient is worked to optimize the 2-bit loss. As a result, the network tends to sacrifices low-precision network accuracy to improve high-precision network accuracy. On the other hand, the separate training, which detaches the gradient from high-precision and trains each precision independently, can achieve the same accuracy as individual training at the lowest precision theoretically. But it limits the potential of models deployed on higher precision.

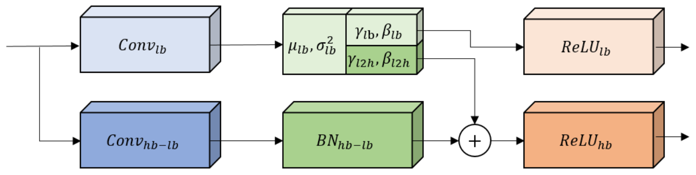
We hope the low precision parameters can take the high precision loss into account but not be constrained by it. To address this, we propose a modified update rule by multiplying the gradient from high precision loss by a scale. The update rule of low precision parameters θ_{lb} is defined as:

$$\theta_{lb}^{n} = \theta_{lb}^{n-1} - \eta(\frac{\partial L_{lb}}{\partial \theta_{lb}} + \lambda \frac{\partial L_{hb}}{\partial \theta_{lb}}), \tag{6}$$

where η is the learning rate and the terms in bracket can be considered as the modified gradient, n is the iteration, λ is the scale for high precision loss. Note it is different from changing the weight α_k of high precision loss in equation 5. The scale λ is only multiplied to the low precision parameters' gradient from high precision loss, and we maintain the high precision parameters' gradient unchanged as $\partial L_{hb}/\partial\theta_{hb}$. It is more flexible in multi-



(a) Vanilla DPOQ


$Conv_{lb}$
μ_{lb}, σ^2_{lb}
γ_{lb}, β_{lb}
$\gamma_{l2h}, \beta_{l2h}$
$ReLU_{lb}$
$Conv_{hb-lb}$
BN_{hb-lb}
+
$ReLU_{hb}$

(b) DPOQ + PSBN

Figure 2: The basic block of a dynamic precision onion quantization (DPOQ) network.

precision conditions, different scales can be assigned to different precision. The gradient scale approach is a combination of joint training and separate training, and the scaling factors of these two training methods are 1 and 0, separately. We set the scale as lb/hb, which is a compromise value in between. It is bit-width related and takes the ability of different precision networks into consideration.

In addition, we do not adopt the self-distillation Zhang et al. (a), which is used in many early exits methods, because we find the effect of self-distillation is not satisfactory, and it may be caused by the particularity of quantization and our network architecture.

3.2. Precision Shift Batch Normalization

How to reuse the information from low-precision is a thorny problem. The element-wise addition used in Figure 2(a)subfigure is not the best option, because the two precision convolutions may have different distributions and contradict each other. Using channel-wise concatenation with bottleneck convolution like DenseNet is more rational, but it increases the number of channels and brings too many extra operations for the high-precision. We reach a balance between cost and performance by using a channel-wise affine transformation, which is also used in the batch normalization (BN) layer Ioffe and Szegedy (2015). The original BN is proposed to reduce internal covariate shift, and the output of the BN layer is defined as follows :

$$BN(y) = \gamma \frac{y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{7}$$

where y is the input feature to be normalized, μ and σ^2 are channel-wise mean and variance of y, ϵ is a small value to prevent the denominator from becoming zero, γ and β are learnable scale and shift. μ and σ^2 are calculated from the current mini-batch during training, and moving average statistics of them are used instead during evaluation. y is normalized to the standard normal distribution $N(0, 1)$ with μ and σ^2 first, and then dose the affine transformation to the desired distribution with learnable scale γ and shift β.

Based on the original BN, we make some modifications and propose our precision shift batch normalization (PSBN) to balance the convolution results from two different precisions. The PSBN has an extra output, which is used to transform the output to the expected

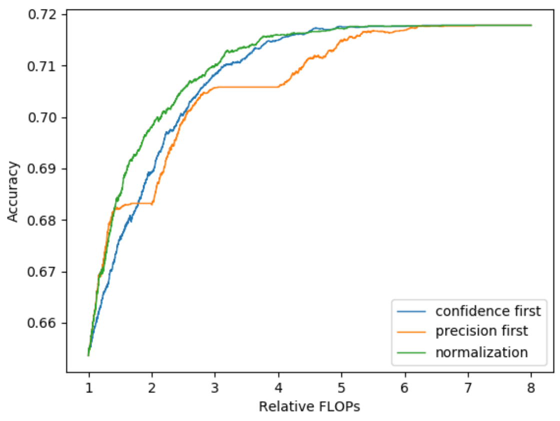


Figure 3: Flops VS. Accuracy. The value of Flops is the relative value to the 2-bit quantization network

distribution with precision-specific γ_{l2h} and β_{l2h}:

$$PSBN_{l2h}(y) = \gamma_{l2h}\frac{y-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta_{l2h}. \tag{8}$$

Instead of performing the element-wise adding operation right after the convolution layer, we push back the calculation of high-precision convolution results to the position after normalization as shown in Figure 2(b)subfigure. It brings more flexibility and better compatibility with each precision. For example, when the gradients from two different precision loss functions conflict to each other, the low-precision convolution parameters in vanilla DPOQ may take a compromised result and perform lower in both precisions. The PSBN can ease the conflicts with multiple sets of scale and shift parameters for each precision. Though the PSBN introduces additional parameters and operations, the cost can be neglected comparing to convolution layers.

3.3. Dynamic Precision Inference Mechanism

Once the network is trained, it can be deployed to adaptive bit-width according to the hardware and resource constraints without further training. And because of our special data reuse mechanism, it can also be seen as a network with multi-exits. We introduce the input as a new variable and quantize the network dynamically according to it.

We use the confidence to decide when to exit. Confidence is the maximum output after softmax, and it is $\in [0, 1]$. If the confidence is bigger than a preset threshold, which means the network is confident enough about the result, then we can exit the network and move on to the next image. Otherwise, we should use a higher precision network to run this image. This procedure is repeated until one sub-network yields a confident prediction, or the highest precision network is utilized. Now, the problem becomes how to select the thresholds for each precision, and it is a trade-off between accuracy and operations. In this section, we will not give out the thresholds directly. Instead, we illustrate how to plot the steepest curve of FLOPs (floating-point operations) vs. accuracy. The user can select an appropriate point on the curve according to demands, and the values of each threshold can be calculated from any point on the curve.

For simplicity, we take the network with 4 precisions (2,4,8 and 16 bits) as an example. First, we run the evaluation of all the images in the lowest 2-bit precision and get the confidence. Then we sort the confidence and start rerunning in a higher 4-bit precision from the least confident image. During the rerunning, we also get confidence from higher precision results. How to treat the confidence from higher precision is a new problem. We design two naive rerunning strategies for that: one is to ignore the confidence from the higher precision, first rerun all the images from the current precision, then move to a higher precision, we call it ”precision first” way; Another is to put the higher precision confidence into the sort list, and rerun the image in the order of confidence, we call it ”confidence first” way. By the way, the thresholds calculated from the ”confidence first” curve are the same for all the precisions.

The result is shown in Figure 3, the higher slope, the higher cost performance. Note that the accuracy is calculated from the ensemble prediction results, otherwise there will be a decline when the threshold gets close to 1. We can see that the two strategies have their strengths and weaknesses. The ”precision first” way exhibits a stepwise increase, but it meets the bottleneck when the confidence threshold is near 1. The ”confidence first” way is more smooth, but the initial slope is not as steep as the ”precision first” way. The problem of the ”confidence first” way is that the FLOPs of rerunning in low precision and in high precision are different, and a distinction should be made for different precision. For example, two images have the same confidence, one confidence is from 2-bit and another is from 4-bit. The 2-bit image has a higher rerunning priority because it consumes less rerun resource. We normalize the high precision confidences before adding them to the low precision confidence sort list. The ”precision first” way can also be treated as a kind of normalization: adding 1 to the confidence of higher precision. But from its results, we can see that the linear normalization can not give consideration to both the low confidence region and the high confidence region. The linear transformation will also change the range of confidence which is undesirable. So we propose our normalized confidence by a power function as:

$$confidence_{norm} = confidence^{lb/hb}, \tag{9}$$

where lb and hb are the number of bits for low and high precision separately. Since the original confidence is $\in [0, 1]$, the value is still in that range after the power function. And when two images have the same original confidence, the network will run low-precision first by using this kind of normalization. From Figure 3 we can see that it combines the advantages of both the ”precision first” way and the ”confidence first” way, and achieves the highest cost performance.

4. Experiments

We evaluate our DPOQNet on the CIFAR and ImageNet classification task and compare the results with individually quantize models. We use the DoReFa Zhou et al. (2016) to quantize the weight, and the PACT Choi et al. (2018) to quantize the activation. We quantize all the convolution layers including the first layer.

Table 1: Ablation results of ResNet20 on CIFAR100 dataset. ”16/fp32” denotes the 16 bit for our DPOQ and the fp32 for the individual training, and the activation bits is 8.

Weight bits	2	4	8	16/fp32	avg
Individual	65.42	68.18	68.61	68.74	67.74
Joint	62.96	68.82	69.13	69.21	67.53
+PSBN	62.63	67.60	70.30	71.05	67.90
Separate	65.30	67.04	67.55	67.49	66.85
+PSBN	65.30	67.36	68.68	69.41	67.69
Grad Scale	64.99	68.20	68.30	68.37	67.47
+PSBN	65.26	69.00	70.32	70.60	68.80

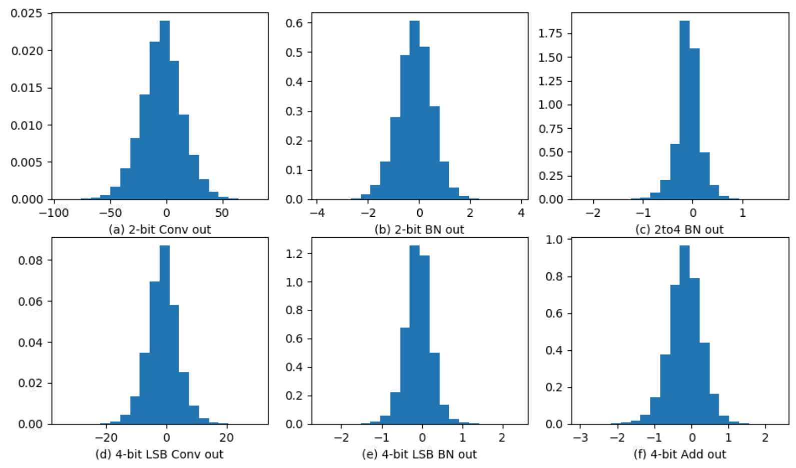


Figure 4: The distributions of feature maps in ResNet20 (layer 13).

4.1. Ablation Experiments

We present an ablation study on how the training modes and the PSBN affect the overall performance. The results are presented in Table 1, and two conclusions are drawn.

Firstly, the results of joint training and separate training are in accordance with our analysis. Joint training even surpasses individual training performance at high precision, while its 2-bit accuracy suffers from an about 3% drop. The effect of separate training is mediocre. It can achieve similar accuracy as individual training at low precision, but its accuracy grows slowly as the bit-width increases. All the training strategies are feasible and the choice primarily depends on which precision the user wants the DPOQ to work better. We use the gradient scale in the rest of this paper, which is more comprehensive and achieves the highest average accuracy.

Secondly, we can see the PSBN improves the performance for the table. With the help of PSBN, some training modes can achieve even higher average accuracy than the individual modes. To further understand the impact of PSBN, we visualize the distributions

Table 2: CIFAR results.

Network	Wbits	Abits	CIFAR10		CIFAR100	
			Individual	DPOQ	Individual	DPOQ
ResNet20	2	2	**88.27**	87.43	**59.45**	58.15
	4		89.84	**90.07**(86.69)	62.40	**63.69**(63.12)
	8		89.98	**91.08**(90.86)	63.35	**67.39**(66.91)
	16/fp32		90.13	91.34(**91.39**)	63.51	68.10(**68.47**)
VGG11(BN)	4	4	92.10	**92.14**	**70.72**	70.64
	5		92.18	**92.46**(92.44)	71.22	**71.26**(71.14)
	6		92.49	**92.57**(92.45)	71.39	**71.49**(71.35)
	7		92.46	**92.58**(92.49)	71.31	**71.47**(71.42)
MobileNetV2	4	8	88.97	**89.03**	68.53	**69.20**
	6		**89.18**	89.05(89.05)	68.70	**69.48**(69.30)
	8		**89.10**	89.06(89.06)	69.28	**69.89**(69.55)

Table 3: ImageNet results.

Network	Bits	Individual	Jin et al. (2,4,6,8)	DPOQ (2,4,6,8)	DPOQ (2,4,8,16)	DPOQ (4,5,6,8)
ResNet18	2	**64.5**	54.1	63.9	64.2	-
	4	**68.7**	66.5	67.7	67.1	68.1
	5	68.9	-	-	-	**69.6**
	6	69.11	67.7	69.5	-	**70.7**
	8	69.6	67.9	69.9	69.6	**70.9**
	16/fp32	69.4	-	-	**70.2**	-

of intermediate feature maps in Figure 4. We find the distributions of two low precision convolution results in (a) and (d) are different, and the direct addition is inappropriate. The distributions of two outputs of PSBN (b) and (c) are also different. The PSBN shifts the 2-bit feature maps to a similar distribution as 4-bit LSB BN layer output (e). The final 4-bit feature map (f) is calculated by adding (c) and (e). The figure and the table both prove that the PSBN is necessary and effective, the affine transformation decreases the distributions difference among different bit-width convolution outputs and assigns more flexibility to the network.

4.2. CIFAR and ImageNet results

The CIFAR dataset contains 50,000 training and 10,000 testing images, and the size of each image is 3 channels (RGB) $\times$ 32 $\times$ 32 pixels. CIFAR-10 and CIFAR-100 contain 10 and 100 categories respectively. We train the network using stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 5e-4. The networks are trained from scratch for 200 epochs with a multi-step learning rate policy. The learning rate is initialized as 0.1 and divided by a factor of 10 at the 100th, 150th and 180th epoch. To make a fair comparison, we train the DPOQ the same way as individual training. Experimental results on CIFAR

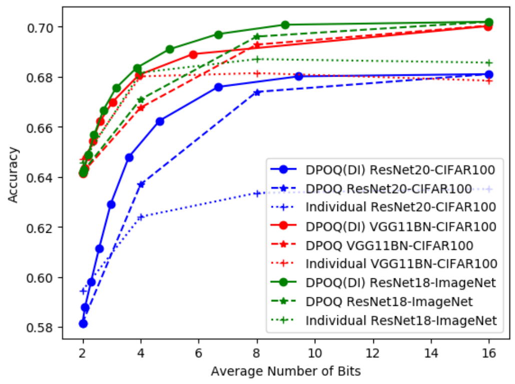


Figure 5: Accuracy of DPOQ, "DI" denotes dynamic inference. (Better view in color.)

dataset are depicted in Table 2. We test several combinations of weight and activation bits on different networks. DPOQ boost accuracy of about 5% on the 16-bit ResNet20 in CIFAR100 dataset. The ImageNet Deng et al. dataset comprised of 1.28M training images and 50K validation images with 1000 classes. The network is trained for 90 epochs in total with a batch size of 256. The network is trained with an SGD optimizer with a momentum of 0.9 and a weight decay of 1e-4. The learning rate is initialized as 0.1 and reduced by 0.1 for every 30 epochs. The results are summarized in Table 3. Although our DPOQ performs not so well in low precision, we improve the upper limit of about 1.5% on 8-bit ResNet18 without increasing the model size.

In summary, the DPOQ shows good compatibility and achieves almost the same performance as individual quantization-aware training for different models and datasets across all precisions. The performance of our DPOQ network is boosted with the sub-networks growth, especially our highest precision network surpasses the full-precision individual network. However, the responsibility of the lowest precision network is overemphasized. It gives consideration to all the other precision loss and may fail to optimize. We put forward some suggestions for the design of adjustable bit list: support fewer kinds of bits, assign more bits to the lower precision network, or use over-parameterized models. Then the high precision loss can be seen as a regularization term and even improve the performance. The effect of the ensemble prediction is mixed. It improves the performance for some models and makes a counter effect for others.We conjecture it is mainly due to the high coincidence degree of two precision outputs. The correctly predicted sample set of the high precision network contains almost all the correctly predicted samples of the low precision network. Then the low precision results are useless and even harmful for the ensemble prediction.

4.3. Dynamic inference results

Experiment results for dynamic inference are depicted in Figure 5. The transverse axis is the average number of bits. The longitudinal axis is the Top1 accuracy evaluated on CIFAR100 and ImageNet dataset. All the DPOQ models are trained in 2,4,8 and 16 bits.

The dotted line with "cross" denotes the baselines of individual training and the dotted line with "star" is the DPOQ without dynamic inference.

It is observed that: the accuracy of our DPOQ fluctuates from low to high compared to individual training. With the same number of bits, our DPOQ with dynamic inference achieves more accurate classification results than without it. The dynamic inference mechanism leads to a faster growth and makes that the accuracy reaches its saturation point earlier. It can make up the accuracy drop in low precision with relatively fewer extra operations.

5. Discussion and Conclusion

In this paper, we propose DPOQ. Previous works have proved that bit-width is an additional degree of freedom for dynamic deployment besides depth, width, kernel size, and so on. In this section, we discuss the advantages and disadvantages of our DPOQ method and make a comparison with other switchable precision methods. The main idea of other precision switchable methods is training a multi-function full-precision weight which can be quantized to different number of bits. They share the same full-precision weights, so we call it the "shared-FP" method for simplicity.

The "shared-FP" method achieves almost the same accuracy as individual quantization for all bit-widths. Our DPOQ method usually has a better performance for high-precision than for low-precision. The highest precision network achieves higher accuracy than the FP32 base line. Our DPOQ method is easier to add a new bit-mode compared to the "shared-FP" method. We support the separate training and we only need to train the additional part of the network and fix the remaining part. While the "shared-FP" method has to train from scratch. Our DPOQ method is more flexible than the "shared-FP" method. Owing to our network architecture and reuse strategy, we develop the input-dependent adaptive inference by dynamically deciding the network precision. DPOQ also further improve the performance.

Acknowledgments

This work is supported by the National Key R&D Program of China (2020YFB0906000, 2020YFB0906001).

References

Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7948–7956.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 3009–3018.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3123–3131.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255.

Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420, 2016.

Sorin Mihai Grigorescu, Bogdan Trasnea, Tiberiu T. Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *J. Field Robotics*, 37(3):362–386, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4107–4115.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 2704–2713. IEEE Computer Society.

Qing Jin, Linjie Yang, and Zhenyu Liao. Adabits: Neural network quantization with adaptive bit-widths. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2143–2153.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3301–3310. PMLR.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1325–1334.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 525–542.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4780–4789.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2464–2469.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pages 420–436.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2366–2375.

Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1803–1811, a.

Jiahui Yu and Thomas S. Huang. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. *CoRR*, abs/1903.11728, b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3712–3721. IEEE, a.

Linfeng Zhang, Zhanhong Tan, Jiebo Song, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. SCAN: A scalable neural networks framework towards compact and efficient models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 4029–4038, b.

Yichi Zhang, Ritchie Zhao, Weizhe Hua, Nayun Xu, G. Edward Suh, and Zhiru Zhang. Precision gating: Improving neural network efficiency with dynamic dual-precision activations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, c.

Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. IEEE Computer Society.