Article
Improving Model Capacity of Quantized Networks with Conditional Computation

Phuoc Pham and Jaeyong Chung *

System on Chips Laboratory, Department of Electronics Engineering, Incheon National University, Incheon 22012, Korea; phuocphn@inu.ac.kr
* Correspondence: jychung@inu.ac.kr

Abstract: Network quantization becomes a crucial step when deploying deep models to the edge devices as it is hardware-friendly, offers memory and computational advantages, but it also suffers performance degradation as the result of limited representation capability. We address this issue by introducing conditional computing to low-bit quantized networks. Instead of using a fixed, single kernel for each layer, which usually does not generalize well across all input data, our proposed method tries to use multiple parallel kernels dynamically in conjunction with the winner-takes-all gating mechanism to select the best one to propagate information. Overall, our proposed method improves upon the prior work, without adding much computational overhead, results in better classification performance on the CIFAR-10 and CIFAR-100 datasets.

Keywords: quantized networks; model compression; dynamic neural network; conditional computing; model capacity; model representation; machine learning; VLSI

1. Introduction

Deep Convolutional Neural Networks (DCNNs) have gained more popularity and achieved promising results in many tasks including computer vision [1], natural language processing [2], reinforcement learning [3], and other broader artificial intelligence fields [4]. In the search for higher performance, recent network architectures [5–8] tend to increase their computational complexity making them more difficult to deploy to resource-constrained devices, where the memory bandwidth, storage, and computation power is much less than the common desktops or cloud computers.

This has motivated the community looking for ways to reduce the model complexity of DCNNs to run efficiently on resource-constrained edge devices. Common techniques include model quantization [9–15], pruning [16–18], low-rank decomposition [19,20], hashing [21], neural architecture search [22,23]. Among these approaches, quantization-based methods have emerged as a promising compression solution and achieved substantial improvements in the recent years. It has been demonstrated that if both the weights and activations are correctly quantized, the expensive convolution operations can be efficiently computed via bitwise operations, enabling fast inference without using Graphics Processing Units (GPUs).

The quantization process maps continuous real values to discrete integers, representing the network weights and activations with very low precision, thus yielding highly compact Deep Neural Network (DNN) models compared to their floating-point counterparts. On the other side, decreasing the bit-width of deep networks naturally brings some noticeable problems such as: performance degeneration [24], training instability [25], and vulnerable to adversarial attack [26].

Lack of representation capability is one of the reasons for the above mentioned problems, and made the quantized networks perform poorly compared to the full-precision counterparts. To improve the model representation capacity, different approximations of quantization and binarization have been studied [13,27,28], mainly categorized into either:
focus on improving the quantization function to reduce the quantization errors, assuming the architecture design is fixed (value approximation) [13,15,28], or try to match the capability of floating-point models by redesigning the architectures (structure approximation). For example, LQ-Net [28] and XNOR-Net [15] increase the model expression ability by relaxing the approximation, using floating-point values to represent the learned basis vectors, and scaling factor of the quantized and binarized weights/activations, respectively. In the category of structure approximation, ABC-Net [29], Group-Net [30] uses multiple bases to approximate the original weight values, with the main purpose of preserving its representability. Unfortunately, the dependence between the bases and input is ignored in these methods, therefore during the forward pass, all input data needs to pass through all bases to form the final layer’s activation, thus making the inference process slower. In other words, although the model performance can be improved significantly by using multiple bases to replicate the original layer, it also introduces more computation overhead and complexity in both training and testing stages.

Another direction to increase the size and representation capacity of a quantized model while maintaining inference efficiency is to use conditional computing [31–35], which aims to increase the model capacity and remain at roughly the same computation cost. In the standard convolution models, a fundamental assumption is that the same configurations (depth, width, bit-width, normalization, activation function, etc.) is applied across to every example in a dataset, which does not generalize well enough. In contrast, for conditional computation models, different configurations are used and only one is selected based on the input example. Previous works were interested in using different convolutional kernels [35], bit-widths [36,37], activation functions [38]. In this scope of this paper, we focus on increasing the model representation by using multiple kernel weights for each layer. Compared to increasing the size of the standard models, this is much computationally efficient to boost representation capability, because only one kernel is used per input, and the cost for kernel aggregation or selection is negligible. Dynamic Convolutional Neural Networks (DY-CNNs) [39] and Conditionally Parameterized Convolutions (CondConv) [40] are the first two attempts to conditionally parameterize the convolution kernels as a linear combination over a set of K parallel experts \( W_1 \), \( W_2 \), \ldots , \( W_k \). The final convolution kernels are formed by aggregating on the fly \( \hat{W} = \sum_{i=1}^{k} \pi(x)_i W_i \) for each input \( x \) and the input dependent attention \( \pi(x) \). However, these works concern about increasing the representation of the floating-point networks only, and do not consider any quantized neural networks. Although quantized networks could benefit from the same dynamic conditional computation idea to boost the final performance, a naive approach of using such strategies is infeasible and not guarantee to get good results as doing simple aggregation will eventually increase the number of discrete values in the quantized networks, leads to using larger bit-widths or re-quantization to compensate for that. In any case, a large computation complexity is required, making it impractical.

The closest work to ours is Expert Binary Convolution [41], where conditional computing is used to implement data-specific expert binary filters and dynamically select a single expert for each sample and use it for processing the input feature maps. A single expert selection strategy is used instead of aggregating multiple kernels into the single one as DY-CNNs and CondConv. In terms of optimization, dynamic networks are more difficult to train and require joint optimization of different convolution kernels and attention modules across many layers. The problem is even challenging in the case of one single expert selection since the routing decisions between individual input data to different experts are discrete choices. To address this issue, the winner-takes-all gating mechanism is applied for the forward pass, and a softmax function is used for gradient approximation of the gating function.

Although Expert Binary Convolution reported the new performance record on model binarization, using multiple binary experts is not only the main factor that leads to higher accuracy, it also relies on exploring better architecture configurations and complex training procedures to match the prediction performance of the floating-point counterpart. Moti-
vated by the promising results of model binarization in Expert Binary Convolution and the current state-of-the-art works in low-bit quantization [12,42], where 4-bit or even 3-bit quantized models already match the full-precision models but at the expense of training complexity or computational overhead. In this paper, we aim to apply the conditional convolution for low-bit quantized networks, dynamically select only kernel weights to process at a time, based upon their input-dependent attentions. Given the same network architecture, the low-bit quantized networks (2, 3, 4 bit) already have higher representation capacity, which is in the middle range between binarized models and full-precision models, and offer a better trade-off between performance and storage and execution time requirements. Therefore, changing/exploring the new architecture design or re-arranging layers is not important as in model binarization, thus we can solely focus on improving the model capacity under quantization constraints. Moreover, we can use bit convolution kernels for any low bit-width networks if it is properly implemented as mentioned in [9]. The blurring of the boundaries between low-bit quantization and model binarization makes the model quantization more appealing when it comes to preserving the floating-point network performance.

We demonstrate our method on the CIFAR-10 and CIFAR-100 classification dataset with ResNet-20, -32. Compared to the previous baseline on low-bit quantization, our quantizer is trained in an end-to-end manner without leveraging any special optimization techniques or architecture modifications, while achieving significantly improvements over the baseline.

2. Related Works

2.1. Model Binarization and Quantization

Since the early work of Courbariaux et al. [14] demonstrated the feasibility of using full binarized weights and activations for inference, and Rastegari [15] reported the very first work archiving high accuracy in the large-scale image classification ImageNet dataset, which uses simple, efficient, and accurate 1-bit approximations for both weights and activations. These works have paved the way to much more advantageous, sophisticated approaches [15,43–46]. It is worth mentioning at, many of these improvements come from relaxing binarization constraint or improving the model representation capacity, for example, XNOR-Net [15] increases the model expression ability by relaxing the binary approximation, in which a scale factor of floating-point is introduced and optimized via back-propagation or calculated analytically. Real2Bin [45] increases the representation power of the convolutional block by combining a gating strategy and progressive attention matching. ABC-Net [29] uses the linear combination of multiple bases for approximating full-precision weights. Similarly, Group-Net [30] decomposes the floating-point network into multiple groups and approximates each group using a set of low-precision bases.

Binarizing the weights and activations to only two stages \{-1, +1\} could lead to a significant loss. To narrow this accuracy gap, ternary neural networks [47,48] and higher bit-width (2-, 3-, 4- up to 8-bit) quantization methods are proposed. Compared to model binarization, these methods require more storage and higher computational complexity, but the accuracy drop can be mitigated if it is trained properly. The early work DoReFa-Net [9] performs a convolution operation in a bit-wise fashion by quantizing both weights and activations as well as gradients with multiple-bits rather than binarize to \{-1, +1\} to further improve accuracy. There has been a great research effort later to improve quantized models including using non-uniform quantization or auxiliary modules [46,49–51], relaxing the discrete optimization problem [13,27], learnable quantizer [10,12,52], mix-precision quantization [11], neural architecture search [23,53]. Noticeably, recent works LSQ-Net [12], QKD [42] even get higher accuracy compared to the full-precision models but at the expense of training cumbersome and computational overhead during inference, which may hinder its applications. We also notice the same pattern for model quantization, as the high correlation between trying to increase model capacity and having training stability and higher final accuracy. LQ-Net [28] uses floating-point values to represent the learned
basis vectors of the quantized weights and activations and minimizing the mean-squared error as the main metric for optimization. In the [50] the model representation capability of the models is enhanced during training thanks to utilizing weight sharing to construct a full-precision auxiliary module. Ref. [54] tries to do progressive quantization, so that the model representation between the quantized models and full-precisions are close to each other at the beginning of the training process, thus make the training more stable.

The recent model quantization work LSQ-Net [12] is a simple but effective method for training efficient neural networks. This is complementary to our work, helping reduce the model size for our dynamic convolution method. Our quantization method aims to obtain low-precision networks with higher representation and generalize well to the input sample without imposing much computational overhead. To archive this, we use conditional computing for low-bit quantization and learn data-specific kernel weights which are selected dynamically during inference based on input data attention.

2.2. Conditional Computations

Our method is related to the dynamic neural network category [31–35] which focus on executing a portion of an existing model conditioned on input data. SkipNet [31], D2NN [55], and BlockDrop [32] use an additional controller for making skipping decisions whether to use convolutional layers or blocks for each input data. Slimmable Nets [56] and US-Nets [57] can work at different model widths. AdaBits [37] and Quantizable DNNs [58] learn a single neural network executable at different bit-widths. Hypernetworks [59] using another network for weight generalization. Once-for-all [60] trains a network that supports multiple sub-networks. Dynamic ReLU [38] parameterized its parameters over all input elements. Dynamic normalization [61] learns arbitrary normalization operations for each convolutional layer. Compared with these works, our method is different in many aspects: Firstly, our method uses dynamic convolution kernels but keeping the network structure static, while previous works have done in the opposite way: having static convolution kernels and letting network structure change. Secondly, we use an embedded attention module in each layer, instead of using an additional external controller, which has the benefit of easy integrating into the existing models.

The proposed methodology in this paper has some relations with previous works: we use LSQ [12] as the quantization baseline method to build upon it. The dynamic convolution as described in Section 4 is somewhat related to dynamic convolutions [39,40], which adapts convolution kernels based on their attentions that are input dependent. However, instead of using multiple full-precision kernel weights, and doing aggregation to form the final weights, we use quantized weights and apply winner-takes-all gating mechanisms similar to [41] to select only one kernel weight for execution. Compared to [39,40], our method tries to work with quantized models, and uses conditional computation to increase the representation ability, thus boosting the final accuracy. Contrary to Expert Binary Convolution, we target low-bit quantized networks, and improve the generalization of convolution layers while keeping the model architectures fixed.

3. Background
3.1. Learnable Step Size Quantization

Following LSQ [12], a symmetric quantization scheme with a learnable step size parameter is used for both weights and activations is defined as follows:

\[
\bar{x} = \left\lfloor \text{clamp}(\frac{x}{s}, Q_N, Q_P) \right\rfloor
\]

\[
\hat{x} = \bar{x} \cdot s
\]

where \(\lfloor\cdot\rfloor\) is the rounding function, \(\text{clamp}(x, Q_N, Q_P)\) is the function clamps all values between an upper bound \(Q_P\) and lower bound \(Q_N\), where values below \(Q_N\) are set to
and values above $Q_P$ are set to $Q_P$. $\bar{x}$ and $\hat{x}$ are the coded bits and quantized values, respectively. Given the bitwidth $b$, for activation (unsigned data), we have:

$$Q_N = 0, Q_P = 2^b - 1$$

and for weights (signed data), we have:

$$Q_N = 2^{b-1}, Q_P = 2^{b-1} - 1$$

During inference, coded bits for activations $\bar{x}$ and coded bits for weights $\hat{w}$ can be used for low precision integer matrix multiplication, and the output is then rescaled by the step size with relatively low computation cost. In order for the gradient to pass through the quantizer, the straight through estimator (STE) is used to approximate the gradients:

$$\frac{\partial \hat{v}}{\partial s} = \begin{cases} 
-\frac{v}{s} + \left \lfloor \frac{v}{s} \right \rfloor & \text{if } -Q_N < \frac{v}{s} < Q_P \\
-Q_N & \text{if } \frac{v}{s} \leq -Q_N \\
Q_P & \text{if } \frac{v}{s} \geq Q_P 
\end{cases}$$

3.2. Dynamic Convolution

In this section, we review the dynamic convolution commonly used in the related literature. Following [39, 40], dynamic convolution with a set of $K$ parallel convolution kernels (or experts) $\hat{W}_k$ are used, instead of using a fixed convolution kernel weight per layer, thus dynamic convolution has more representation power compared to its static counterpart. The final convolution kernel is the output of a nonlinear function, which is dynamically aggregated by the following formula:

$$\hat{W} = \sum_k \pi_k(x) \hat{W}_k$$

where $x$ is the input, and $\pi_k(\cdot)$ is the attention gating module via input dependent. The attention module can be made from convolution, linear and other layers. In CondConv this function is composed from three steps: global average pooling, fully connected layer and sigmoid activation.

$$\pi(x) = \text{Sigmoid}(\text{FC}(\text{GlobalAveragePooling}(x)))$$

In [39] the squeeze-and-excitation strategy derived from [62] is used instead. First, the global average pooling is used to squeezed the global spatial information, and then two fully connected layers with a non-linear activation function are used to create the intermediate results. Finally, the intermediate results passed through the softmax layer to generate the normalized attention weights for $K$ convolution kernels. More details about the module setting can be found in [39].

$$\pi(x) = \text{Softmax}(\text{FC2}(\text{ReLU}(\text{FC1}(\text{GlobalAveragePooling}(x)))))$$

Obviously, the later attention design is heavier computation. Having a suitable, computationally efficient configuration is a desire for this module as it will be used repeatedly during inference. Different designs for the attention modules will be evaluated and discussed in detail in the Section 5.

4. Dynamic Quantized Convolution

For a convolutional layer, we define the input $x \in \mathbb{R}^{C_{in} \times W_{in} \times H_{in}}$, weight filter $W \in \mathbb{R}^{C_{in} \times C_{out} \times k_W \times k_H}$ and the output $y \in \mathbb{R}^{C_{out} \times W_{out} \times H_{out}}$, respectively. $C_{\{in, out\}}$ denotes the number input and output channels, $W_{\{in, out\}}, H_{\{in, out\}}$ denotes the width, height of input and output feature maps, while $k_W, k_H$ denotes the kernel size. In normal convolution, the weight filters are fixed and used for all input samples. In contrast, we use
a set of $K$ parallel learnable kernel weights (or experts) $\{W_0, W_1, W_2, \ldots, W_{K-1}\}$ with the same dimension as in the original convolution and stacked them to form a matrix: $W \in \mathbb{R}^{K \times C_{in} \times H \times W}$. Given input $x$, the attention module together with a gating function outputs the attention over convolution kernels and select the one with highest probability.

We define the dynamic quantized convolution (DQConv) as follow:

$$
DQConv(x, \hat{W}) = \text{Conv}(Q(x), \alpha_1 Q(W_1) + \alpha_2 Q(W_2) + \ldots + \alpha_n Q(W_n))
$$

$$
= \text{Conv}(Q(x), \phi(\pi(x))^T Q(\hat{W}))
$$

$$
= y
$$

(8)

where each $\alpha_i = \phi(\pi(x)_i)$, and $\alpha_i \in \{0, 1\}$ plays the role of switches to determine which convolution kernel is used for the later convolution operations. $\pi(\cdot)$ can be implemented either as Equation (6) or Equation (7), $\phi(\cdot)$ is the gating function, which will be described detail below, $Q(\cdot)$ is the quantization function, which is implemented follows Learned Step Size Quantization (LSQ) in this paper, and Conv($\cdot$) is the convolution operator and it is supported in most of the deep learning frameworks. For simplicity, the bias in convolution is ignored. The overview of DQConv is depicted in the Figure 1.

By doing so, it also introduces a new hyper-parameter for tuning, the number of experts will be used for each convolution layer ($K$). Increasing the number of experts will also increase the model representation capacity but also slower the training process. Finding an optimal number of experts is crucial for deployment. The investigation about this trade-off issue is done in the experiment section.

**Gating function $\phi(\cdot)$ and Attention module $\pi(x)$**: Inspired by Expert Binary Convolution, we use Winners-Take-All function (WTA) for expert selection. For the forward pass, the WTA function is used for $\phi(\cdot)$ as defined as follows:

$$
\phi(x) = \begin{cases} 
  1 & \text{if } i = \text{argmax}(x) \\
  0 & \text{otherwise}
\end{cases}
$$

(9)

This function returns a $K$-dimensional vector, and the Hadamard product between its outputs and stacked matrix $\hat{W}$ will be used for the later convolution operation. Note that, compared to CondConv, which uses softmax function and aggregating over kernels. The WTA function is not differentiable and can not back-propagate gradients during train-
ing, therefore for the backward pass, softmax function is used for gradient approximation of function $\phi(\cdot)$ as it can effectively address the gradient mismatch problem and also allow the gradients pass through non-selected experts during training.

We use two attention module variants from CondConv [40] and DY-CNNs [39] for evaluation. The two structures are summarized by Equations (6) and (7). In overview, the former is much lighter in terms of computation complexity, while the later can better capture the discrimination of input samples. We will also examine the effectiveness of each configuration in the Section 5.

**Softmax temperature.** Using the temperature $\tau$ in softmax is known to have training stability. We also explore the impact of using a temperature to the softmax function in the attention module. The softmax with temperature $\tau$ is defined as follows:

$$
\pi_k = \frac{\exp(z_k / \tau)}{\sum_j \exp(z_j / \tau)}
$$

where $z_k$ is the output of the last layer in the attention module.

In Expert Binary Convolution, the authors reported the constant softmax temperature $\tau = 1$ offers the best accuracy during stage 1 training (which means no temperature involved). While DY-Conv suggests having near-uniform attention in early training is crucial, and the temperature $\tau$ should linearly reduce from 30 to 1 in the first 10 epochs. Both settings will be considered in this paper and evaluated in the next section.

5. Experiment Results

To demonstrate the effectiveness of our dynamic quantized convolution, we evaluated it on the CIFAR-10 and CIFAR-100 [63] datasets with ResNet-20 and ResNet-32, respectively. The CIFAR10 dataset contains 60,000 $32 \times 32$ color images of 10 different mutually exclusive classes. The CIFAR100 is similar but contains images from 100 classes.

**Implementation details:** We implement our method using PyTorch, and do evaluations with 4 GeForce GTX 1080 GPUs. We use the original pre-activation ResNet architecture [5] without any structural modifications for ResNet-20 and -32. In all the experiments, the training images are kept in its original size of $32 \times 32$, and horizontally flipped at random. We normalize the training and testing images by the mean and standard deviation calculated from the whole dataset. No further data augmentation is used for testing. We use stochastic gradient descent with the batch size of 128, momentum of 0.9, and the weight decay of $5 \times 10^{-4}$. We trained each network up to 200 epochs where the learning rate is initially set to 0.1 and cosine learning rate decay without restarts is used during training.

Inspired by Expert Binary Convolution, we first train these networks with 1 expert only ($K = 1$), and then replicate it to all other convolution kernels $W_i$ to initialize the stacked matrix $\hat{W}$ in the Equation (8). The softmax temperature is set to $\tau = 0.5$ for all experiments, unless mentioned otherwise. For quantized networks, the learning rate of 0.01 is used, and no weight decay is used for learnable step size parameters in LSQ quantizer.

Following the previous works, the first and the last layers are quantized to 8-bit for hardware-friendly and computational efficiency.

**Comparison with LSQ baseline:** We evaluate our dynamic quantization method, denoted as DQConv, with the LSQ baseline. The results are shown in Table 1. In both datasets and different bit-width configurations, we observed the accuracy improvements. Our 4/4-bit models outperform the accuracy of full-precision models and the LSQ baselines for all two comparing network architectures (ResNet-20, -32). For 3/3-bit models, the accuracy of DQConv surpass the full-precision counterparts in both datasets, while the accuracy drops by 0.03% (CIFAR-100) and 0.32% (CIFAR-10) if only using LSQ. The significant impacts can be seen in 2/2 bit models, which 2.3% (CIFAR-100) and 0.99% (CIFAR-10) improvements over the LSQ baselines.
Table 1. Comparison between the dynamic quantized convolution (DQConv) with the LSQ (Learned Step Size Quantization) baselines on CIFAR-10 and CIFAR-100 using ResNet-20 and ResNet-32, respectively. ‘FP’ represents the full-precision 32-bit accuracy in our implementation. The bold numbers indicate the best results.

| Methods          | Bit-Width (W/A) | 2/2      | 3/3      | 4/4      |
|------------------|-----------------|----------|----------|----------|
|                  | LSQ             | 90.83    | 92.54    | 92.85    |
| CIFAR10 (FP: 92.86) | DQConv (K = 3) | 91.82    | 92.89    | 93.07    |
|                  | DQConv (K = 4)  | 91.76    | 92.96    | 93.03    |
| CIFAR-100 (FP: 71.35) | LSQ             | 68.26    | 71.32    | 71.49    |
|                  | DQConv (K = 3)  | 70.41    | 71.98    | 72.68    |
|                  | DQConv (K = 4)  | 70.56    | 71.65    | 72.38    |

Attention structures: Two attention structure variants will be considered and evaluated in this sub-section. Clearly, a simple implementation of attention module with only one or two fully-connected layers, in conjunction with global averaging pooling to reduce the spatial information and a softmax functions works well in most of the cases as reported with the floating-point networks [39,40] and binarized networks [41]. However, the different behaviors and activation distribution of quantized models require more extensive investigations about the optimal structure for the attention module. Over-parameterization could lead to performance degradation, while under-parameterization could make the module struggle to differentiate between different input samples. Table 2 shows the classification accuracy of quantized models with different attention variants. We find that a deeper attention module as Equation (7) is needed for the quantized neural networks, instead of the simple structure as in Expert Binary Convolution and CondConv. There are 0.62%, 0.49%, 0.63% performance improvements for 2-, 3- and 4-bit quantized networks, respectively on CIFAR-100 dataset when switch to the deeper attention structure. However, if preserving the computation efficiency is the important metric, using the simple architecture as Equation (6) will be a better choice as it can offer a better accuracy and computation speed trade-off.

Table 2. The impacts of using different attention structures for quantized neural networks. We consider two attention variants from Equations (6) and (7) for comparison. All experiments are evaluated with the number of experts $K = 3$. The bold numbers indicate the best results.

| Dataset         | Attention Structure   | Bit-Width (W/A) |
|-----------------|-----------------------|-----------------|
|                 |                       | 2/2      | 3/3      | 4/4      |
| CIFAR-10        | Equation (6)          | 91.88    | 92.72    | 92.94    |
|                 | Equation (7)          | 91.82    | 92.89    | 93.07    |
| CIFAR-100       | Equation (6)          | 69.79    | 71.49    | 72.05    |
|                 | Equation (7)          | 70.41    | 71.98    | 72.68    |

The number of convolution kernels ($K$): The model complexity is controlled by the hyper parameter $K$, and it plays a crucial role in achieving the optimal model, which offers good trade-off between model size and computation cost. Table 3 and Figure 2 show the prediction accuracy versus computational complexity for dynamic convolution with different $K$ values. We compare the dynamic quantized and the static quantized versions of ResNet-32 with different number of experts ($K = 2, 3, 4, 5$). From the experiment results, we can observe that the dynamic convolution outperforms its static counterpart for all bit-widths, even if only two experts ($K = 2$) are used. This demonstrates the advantage of using the dynamic convolution layers. It is interesting to note that the accuracy stops increasing when $K$ is larger than 3, especially in case of larger bit-widths are used. The accuracy of
$W_4A_4$ and $W_3A_3$ quantized models stop increasing and even slightly drop with the 4-bit quantized network. As $K$ increases, although the model has more representation power, but optimizing all convolution kernels and attention simultaneously is more challenging and also the degradation in performance can be experienced due to over-fitting.

Table 3. Comparison on CIFAR-100 between the dynamic quantized convolution (DQConv) with different number of experts and the LSQ (Learned Step Size Quantization) baseline. The results suggest $K = 3$ offers the best trade-off among these comparing settings. The bold numbers indicate the best results.

| Bit-Width (W/A) | 2/2 | 3/3 | 4/4 |
|----------------|-----|-----|-----|
| LSQ (Baseline) | 68.26 | 71.32 | 71.49 |
| DQConv ($K = 2$) | 69.92 | 71.21 | 72.04 |
| DQConv ($K = 3$) | 70.41 | **71.98** | **72.68** |
| DQConv ($K = 4$) | 70.56 | 71.65 | 72.38 |
| DQConv ($K = 5$) | 70.21 | 71.5 | 71.89 |

Figure 2. The accuracy of quantized ResNet-32 on CIFAR-100 with the different number of experts.

Quantized vs. Full-Precision Attention Module: Different to floating-point networks, introducing additional full-precision layers to the quantized networks would increase the hardware design burden or even impractical to deploy the model in some edge devices. Therefore, in order to be hardware-friendly and increase the deployment feasibility, we try to quantize all attention modules and other convolution and linear layers in the original networks to low-bit precision. As the attention modules are more sensitive to the input data, and the assumption 8-bit quantization can preserve the performance of the full-precision networks as mentioned in [12], we consider quantizing all fully-connected layers in attention modules to 8-bit precision. The impacts to the final classification performance of these models after doing quantization are shown in Table 4. We can notice that the 8-bit quantized version of these attention modules works well for most quantized models with a small drop in accuracy. It suggests that there is no need to use floating-point numbers to represent the weights in the attention modules and the design burden for kernel implementation can be avoided.

Table 4. The comparison between using the full-precision and quantized attention module. All models are trained with ResNet-32, $K = 3$ on CIFAR-100. The bold numbers indicate the best results.

| Configurations | Bit-Width (W/A) |
|----------------|----------------|
|                | 2/2 | 3/3 | 4/4 |
| DQConv ($K = 3$) + Full-Precision Attention | 70.41 | **71.98** | **72.68** |
| DQConv ($K = 3$) + Quantized Attention | 70.14 | 71.64 | 71.89 |
Softmax Temperature: As mentioned earlier, the softmax temperature used in the attention modules also impacts the training dynamics and efficiency. Adding temperature into softmax will change the final probability distribution. Higher temperature produces a softer probability distribution for expert selection, while low temperature makes the output distribution more sparse. We inspect the effectiveness of different ways of setting softmax temperature. In Expert Binary Convolution [41], the softmax temperature is fixed to $\tau = 1$ for the entire training process. In DY-CNNs [39], the temperature is initially set to 30 and decreases to 1 linearly over the first 10 epochs. We evaluate both approaches in this sub-subsection and the results are shown in Table 5. We find that, using a static, low softmax temperature value (e.g., $\tau = 0.5$) is needed to get the best accuracy, while the temperature annealing as DY-CNNs only attains sub-optimal solutions.

Table 5. The accuracy of quantized ResNet-32 on CIFAR-100 with different configurations of softmax temperature. All experiments are tested with ResNet-32 on CIFAR-100 dataset, with the number of experts $K = 3$. The bold numbers indicate the best results.

| Configurations            | Bit-Width (W/A) |
|---------------------------|-----------------|
|                           | 2/2             | 3/3  | 4/4  |
| DQConv ($\tau = 0.02$)    | 68.77           | 70.93| 71.6 |
| DQConv ($\tau = 0.5$)     | **70.41**       | 71.98| **72.68** |
| DQConv ($\tau = 1.0$)     | 70.18           | 71.75| 71.97|
| DQConv ($\tau = 5.0$)     | 69.83           | 71.56| 72.07|
| DQConv ($\tau = 30$)      | 69.71           | 71.54| 72.15|
| DQConv ($\tau = 30 \rightarrow 1$) | 70.27 | 71.76| 72.33|
| DQConv ($\tau = 30 \rightarrow 0.5$) | 69.64 | 71.41| 72.07|

6. Analysis

6.1. Memory Usage and Run-Time Analysis

We compare different settings of the dynamic quantized convolutional networks in terms of model storage size and run-time complexity. As shown in Table 6, compared to the LSQ baseline, our method required more space to store the parallel convolution kernels. For a ResNet-32 quantized model trained on CIFAR-100 with $K = 3$, it needs from 369.6 to 719.0 kilobytes for storing the weight parameters. Meanwhile, less than 252 kilobytes are needed for LSQ-based quantized networks. It is obvious that increasing the number of experts will result in more space being taken, but this is still negligible. Going in more details, even if we use 5 parallel experts ($K = 5$) with the 4-bit quantized networks, the storage requirements are still less than the original floating-point models, while it offers better accuracy improvements.

In terms of run-time overhead, introducing dynamic quantized convolution layers only increase 0.01 M multiply-accumulate operations (MACs) compared to the floating-point networks. Moreover, if the simple structure for the attention module as Equation (7) is used, there is no significant computation overhead between the LSQ baselines and the full-precision networks can be noticed.

6.2. Attention Visualization

To get more insights about the attention module and the dynamic behavior of DQConv layers, we visualize the attention distribution of different layers in the ResNet-32 with the different number of experts $K = 2, 3, 4$. The histogram graphs are shown in Figure 3. It is interesting to note that, in most cases, earlier layers seem to favor one particular expert and pay little attention to the others, while the later layers show better uniform distribution over experts. But the contribution of each expert (or distribution probability over experts) is different between layers and bit widths. In some layers, only a few experts are selected frequently, while the remaining experts are not used most of the time. Moreover, the distribution also tends to become non-uniform in the higher bit-width networks (3-,
4-bit) compared to the 2-bit quantized model. Therefore to avoid storage redundancy and reduce computation overhead, such layers need special treatments such as using simply fixed convolution layers, or further pruning to obtain the optional model. However, such investigations are beyond the scope of this paper and are the subject of future works.

Figure 3. The attention output distribution over 2 experts (a), 3 experts (b), and 4 experts (c) of different layers in the 4-bit quantized ResNet-32. The y-axis denotes the selection probability.

Table 6. The comparison of the dynamic quantized convolutional networks with different configurations. The key comparison metrics include: The total number of parameters (# Params), model size in Kilobyte, compression rate (CR), and the number of multiply-accumulate operations or MACs (unit: million operations).

| Method      | Bit-Width (W/A) | Attention Structure | # Params | Model Size (KB) | CR   | MACs (M) |
|-------------|-----------------|---------------------|---------|-----------------|------|----------|
| FP32        | 32/32           | -                   | 470,292 | 1790            | 1×   | 69.13    |
| LSQ         | 2/2             | -                   | 470,356 | 137             | 13.06× | 69.13    |
| DQConv (K = 2) | 2/2              | Equation (6)       | 936,056 | 253.5           | 7.00× | 69.13    |
|             | 2/2             | Equation (7)       | 948,360 | 256.5           | 6.97× | 69.14    |
|             | 3/3             | Equation (6)       | 936,056 | 367.2           | 4.87× | 69.13    |
|             | 3/3             | Equation (7)       | 948,360 | 371.8           | 4.81× | 69.14    |
| DQConv (K = 3) | 4/4              | Equation (6)       | 936,056 | 481.0           | 3.72× | 69.13    |
|             | 4/4             | Equation (7)       | 948,360 | 487.1           | 3.61× | 69.14    |
|             | 2/2             | Equation (6)       | 1,400,600 | 369.6           | 4.84× | 69.13    |
|             | 2/2             | Equation (7)       | 1,412,096 | 372.5           | 4.80× | 69.14    |
|             | 3/3             | Equation (6)       | 1,400,600 | 541.4           | 3.30× | 69.13    |
| DQConv (K = 4) | 3/3              | Equation (7)       | 1,412,096 | 545.7           | 3.28× | 69.14    |
|             | 4/4             | Equation (6)       | 1,400,600 | 713.2           | 2.50× | 69.13    |
|             | 4/4             | Equation (7)       | 1,412,096 | 719.0           | 2.48× | 69.14    |
|             | 2/2             | Equation (6)       | 1,865,144 | 485.7           | 3.68× | 69.13    |
|             | 2/2             | Equation (7)       | 1,875,832 | 488.45          | 3.66× | 69.14    |
|             | 3/3             | Equation (6)       | 1,865,144 | 715.6           | 2.50× | 69.13    |
| DQConv (K = 5) | 3/3              | Equation (7)       | 1,875,832 | 719.6           | 2.48× | 69.14    |
|             | 4/4             | Equation (6)       | 1,865,144 | 945.5           | 1.89× | 69.13    |
|             | 4/4             | Equation (7)       | 1,875,832 | 950.9           | 1.88× | 69.14    |
|             | 2/2             | Equation (6)       | 2,329,688 | 601.9           | 2.97× | 69.13    |
|             | 2/2             | Equation (7)       | 2,339,568 | 604.3           | 2.96× | 69.14    |
|             | 3/3             | Equation (6)       | 2,329,688 | 889.8           | 2.01× | 69.13    |
|             | 3/3             | Equation (7)       | 2,339,568 | 893.5           | 2.00× | 69.14    |
|             | 4/4             | Equation (6)       | 2,329,688 | 1177.0          | 1.52× | 69.13    |
|             | 4/4             | Equation (7)       | 2,339,568 | 1182.7          | 1.51× | 69.14    |
6.3. Training Complexity

We would like to highlight that the training complexity is not much different compared to the LSQ baselines. The new proposed layer (DQConv) with the fixed temperature can be used as a drop-in replacement for most network architectures. And the training procedure is identical to the LSQ method. In terms of memory requirement, a large proportion of memory is used for storing the intermediate results for backpropagation, while the kernel weights are only accumulated for a small proportion. Thus, increasing the number of experts only requires small additional memory as mentioned in the second answer. Therefore, we can argue that introducing DQConv to deep neural networks does not increase much training complexity and resource requirements over the baseline method (LSQ).

7. Conclusions

In this paper, we introduced dynamic quantized convolution, which improves the model capacity of the prior state-of-the-art work about low-bit quantization by using multiple experts and select a single expert at a time based on their attention for each input. Compared to the baseline about low-bit quantization, our proposed method can significantly improve the representation capacity with a few extra computation costs. Compared to the floating-point network counterpart, our method even achieves higher accuracy with the same computation budget and requires less storage. In addition, we analyzed different attention module structures and training schemes to find the best configuration for low-bit quantized networks. We found that a deeper attention module and a proper softmax temperature setting are needed for quantized networks. Moreover, attention modules can be further quantized to 8-bit precision without introducing much accuracy loss. We believe that the proposed dynamic quantized convolution layer could be used as a drop-in replacement for most of the existing CNN architectures, thus enabling efficient inference.

Author Contributions: Conceptualization, P.P. and J.C.; methodology, P.P. and J.C.; software, J.C.; validation, P.P.; formal analysis, P.P.; investigation, P.P.; resources, J.C.; data curation, P.P.; writing—original draft preparation, P.P.; writing—review and editing, P.P. and J.C.; visualization, P.P.; supervision, J.C.; project administration, J.C.; funding acquisition, J.C. Both authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Incheon National University Grant in 2018 and the Institute for Information and Communications Technology Promotion funded by the Korea Government under Grant 1711073912.

Acknowledgments: The authors also would like to thank the reviewers and editors for their reviews of this research.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- CondConv: Conditionally Parameterized Convolutions [40]
- DNN: Deep Neural Network
- DCNNs: Deep Convolutional Neural Networks
- DY-CNNs: Dynamic Convolutional Neural Networks [39]
- DQConv: Dynamic Quantized Convolution
- GPU: Graphics Processing Unit
- LSQ: Learned Step Size Quantization [12]
- WTA: Winners-Take-All function
References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]

2. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv 2018*, arXiv:1810.04805.

3. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv 2013*, arXiv:1312.5602.

4. Senior, A.W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A.W.; Bridgland, A.; et al. Improved protein structure prediction using potentials from deep learning. *Nature 2020*, 577, 706–710. [CrossRef] [PubMed]

5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016*; pp. 770–778.

6. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th *USENIX* Symposium on Operating Systems Design and Implementation* (OSDI) *16*, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.

7. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv 2014*, arXiv:1409.1556.

8. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017*; pp. 4700–4708.

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

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

11. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/ICVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019*; pp. 8612–8620.

12. Esser, S.K.; McKinstry, J.L.; Bablani, D.; Appuswamy, R.; Modha, D.S. Learned step size quantization. *arXiv 2019*, arXiv:1902.08153.

13. Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; Yan, J. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/ICVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019*; pp. 4852–4861.

14. Courbariaux, M.; Bengio, Y.; David, J.P. Binaryconnect: Training deep neural networks with binary weights during propagations. *arXiv 2015*, arXiv:1511.00363.

15. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016*; Springer: Cham, Switzerland, 2016; pp. 525–542.

16. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv 2015*, arXiv:1510.00149.

17. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017*; pp. 1389–1397.

18. Luo, J.H.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017*; pp. 5058–5066.

19. Lee, D.; Kwon, S.J.; Kim, B.; Wei, G.Y. Learning low-rank approximation for cnns. *arXiv 2019*, arXiv:1905.10145.

20. Long, X.; Ben, Z.; Zeng, X.; Liu, Y.; Zhang, M.; Zhou, D. Learning sparse convolutional neural network via quantization with low rank regularization. *IEEE Access 2019*, *7*, 51866–51876. [CrossRef] [PubMed]

21. Chen, W.; Wilson, J.; Tyree, S.; Weinberger, K.; Chen, Y. Compressing neural networks with the hashing trick. In *Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015*; pp. 2285–2294.

22. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv 2016*, arXiv:1611.01578.

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

24. Ding, R.; Chin, T.W.; Liu, Z.; Marculescu, D. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE/ICVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019*; pp. 11408–11417.

25. Lin, J.; Gan, C.; Han, S. Defensive quantization: When efficiency meets robustness. *arXiv 2019*, arXiv:1904.08444.

26. Yang, J.; Shen, X.; Xing, J.; Tian, X.; Li, H.; Deng, B.; Huang, J.; Hua, X.S. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Munich, Germany, 8–14 September 2018*; pp. 365–382.

27. Zhang, D.; Yang, J.; Ye, D.; Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018*; pp. 365–382.
30. Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; Reid, I. Structured binary neural networks for accurate image classification and semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 413–422.

31. Wang, X.; Yu, F.; Dou, Z.Y.; Darrell, T.; Gonzalez, J.E. Skipnet: Learning dynamic routing in convolutional networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 409–424.

32. Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L.S.; Grauman, K.; Ferris, R. Blockdrop: Dynamic inference paths in residual networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8817–8826.

33. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv 2017, arXiv:1701.06538.

34. Mullapudi, R.T.; Mark, W.R.; Shazeer, N.; Fatahalian, K. Hydranets: Specialized dynamic architectures for efficient inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8080–8089.

35. Chen, Z.; Li, Y.; Bengio, S.; Si, S. You look twice: Gatedfnet for dynamic filter selection in cnns. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 9172–9180.

36. Guerra, L.; Zhuang, B.; Reid, I.; Drummond, T. Switchable Precision Neural Networks. arXiv 2020, arXiv:2002.02815.

37. Jin, Q.; Yang, L.; Liao, Z. Adabits: Neural network quantization with adaptive bit-widths. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2146–2156.

38. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic relu. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Cham, Switzerland, 2020; pp. 351–367.

39. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic convolution: Attention over convolution kernels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11030–11039.

40. Yang, B.; Bender, G.; Le, Q.V.; Ngiam, J. Condconv: Conditionally parameterized convolutions for efficient inference. arXiv 2019, arXiv:1904.04971.

41. Bulat, A.; Martinez, B.; Tzimiropoulos, G. High-Capacity Expert Binary Networks. arXiv 2020, arXiv:2010.03558.

42. Kim, J.; Bhalgat, Y.; Lee, J.; Patel, C.; Kwak, N. Qkd: Quantization-aware knowledge distillation. arXiv 2019, arXiv:1911.12491.

43. Bulat, A.; Tzimiropoulos, G. Xnor-net++: Improved binary neural networks. arXiv 2019, arXiv:1909.13863.

44. Martinez, B.; Yang, J.; Bulat, A.; Tzimiropoulos, G. Training binary neural networks with real-to-binary convolutions. arXiv 2020, arXiv:2003.11355.

45. Kim, H.; Kim, K.; Kim, J.; Kim, J. BinaryDuo: Reducing Gradient Mismatch in Binary Activation Network by Coupling Binary Activations. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.

46. He, X.; Mo, Z.; Cheng, K.; Xu, W.; Hu, Q.; Wang, P.; Liu, Q.; Cheng, J. Proxibnn: Learning binarized networks via proxy matrices. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Volume 2, pp. 7531–7542.

47. Li, F.; Zhang, B.; Liu, B. Ternary weight networks. arXiv 2016, arXiv:1605.04711.

48. Alemdar, H.; Leroy, V.; Prost-Boucle, A.; Pétrot, F. Ternary neural networks for resource-efficient AI applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2547–2554.

49. Li, Y.; Dong, X.; Wang, W. Additive Powers-of-Two Quantization: A Non-uniform Discretization for Neural Networks. In Proceedings of the International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.

50. Zhuang, B.; Liu, L.; Tan, M.; Shen, C.; Reid, I. Training quantized neural networks with a full-precision auxiliary module. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1488–1497.

51. Zhuang, B.; Liu, L.; Tan, M.; Shen, C.; Reid, I. Training quantized network with auxiliary gradient module. arXiv 2019, arXiv:1903.11236.

52. Jung, S.; Son, C.; Lee, S.; Son, J.; Han, J.J.; Kwak, Y.; Hwang, S.J.; Choi, C. Learning to quantize deep networks by optimizing quantization intervals with task loss. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4350–4359.

53. Chen, Y.; Meng, G.; Zhang, Q.; Zhang, X.; Song, L.; Xiang, S.; Pan, C. Joint neural architecture search and quantization. arXiv 2018, arXiv:1811.09426.

54. Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; Reid, I. Towards effective low-bitwidth convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7920–7928.

55. Dou, H.; Deng, Y.; Yan, T.; Wu, H.; Lin, X.; Dai, Q. Residual D2NN: Training diffractive deep neural networks via learnable light shortcuts. Opt. Lett. 2020, 45, 2688–2691. [CrossRef]

56. Yu, J.; Yang, L.; Xu, N.; Yang, J.; Huang, T. Slimmable neural networks. arXiv 2018, arXiv:1812.08928.

57. Yu, J.; Huang, T.S. Universally slimmable networks and improved training techniques. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 1803–1811.

58. Du, K.; Zhang, Y.; Guan, H. From Quantized DNNs to Quantizable DNNs. arXiv 2020, arXiv:2004.05284.

59. Ha, D.; Dai, A.; Le, Q.V. Hypernetworks. arXiv 2016, arXiv:1609.09106.
60. Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; Han, S. Once for All: Train One Network and Specialize it for Efficient Deployment. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.

61. Luo, P.; Zhanglin, P.; Wenqi, S.; Ruimao, Z.; Jiamin, R.; Lingyun, W. Differentiable dynamic normalization for learning deep representation. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; PMLR; pp. 4203–4211.

62. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.

63. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada, 2009.