Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation

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

Abstract

In this paper, we propose to train convolutional neural networks (CNNs) with both binarized weights and activations, leading to quantized models specifically for mobile devices with limited power capacity and computation resources. Previous works on quantizing CNNs seek to approximate the floating-point information using a set of discrete values, which we call value approximation, but typically assume the same architecture as the full-precision networks. However, we take a novel "structure approximation" view for quantization—it is very likely that a different architecture may be better for best performance. In particular, we propose a "network decomposition" strategy, termed Group-Net*, in which we divide the network into groups. In this way, each full-precision group can be effectively reconstructed by aggregating a set of homogeneous binary branches. In addition, we learn effective connections among groups to improve the representational capability. Moreover, the proposed Group-Net shows strong generalization to other tasks. For instance, we extend Group-Net for highly accurate semantic segmentation by embedding rich context into the binary structure. Experiments on both classification and semantic segmentation tasks demonstrate the superior performance of the proposed methods over various popular architectures. In particular, we outperform the previous best binary neural networks in terms of accuracy and major computation savings.*

1. Introduction

Designing deeper and wider convolutional neural networks has led to significant breakthroughs in many machine learning tasks, such as image classification [17,26], object detection [40,41] and object segmentation [7,34]. However, accurate deep models often require billions of FLOPs, which makes it infeasible for deep models to run many real-time applications on resource constrained mobile platforms. To solve this problem, many existing works focus on network pruning [18,28,54], low-bit quantization [24,53] and/or efficient architecture design [8,21]. Among them, the quantization approaches represent weights and activations with low bitwidth fixed-point integers, and thus the dot product can be computed by several XNOR-popcount bitwise operations. The XNOR of two bits requires only a single logic gate instead of using hundreds units for floating point multiplication [10,14]. Binarization [22,39] is an extreme quantization approach where both the weights and activations are represented by a single bit, either $+1$ or -1. In this paper, we aim to design highly accurate binary neural networks (BNNs) from both the quantization and efficient architecture design perspectives.

Existing quantization methods can be mainly divided into two categories. The first category methods seek to design more effective optimization algorithms to find better local minima for quantized weights. These works either introduce knowledge distillation [35,38,53] or use loss-aware objectives [19, 20]. The second category approaches focus on improving the quantization function [4, 48, 51]. To maintain good performance, it is essential to learn suitable mappings between discrete values and their floating-point counterparts . However, designing quantization function is highly non-trivial especially for BNNs, since the quantization functions are often non-differentiable and gradients can only be roughly approximated.

The above two categories of methods belong to *value approximation*, which seeks to quantize weights and/or activations by preserving most of the representational ability of the original network. However, the value approximation approaches have a natural limitation that it is merely a local approximation. Moreover, these methods often lacks of adaptive ability to general tasks. Given a pretrained model on a specific task, the quantization error will inevitably occur and the final performance may be affected.

In this paper, we seek to explore a third category called *structure approximation* . The main objective is to redesign a binary architecture that can directly match the capability of a floating-point model. In particular, we propose a Struc-

*C. Shen is the corresponding author.

tured Binary Neural Network called Group-Net to partition the full-precision model into groups and use a set of parallel binary bases to approximate its floating-point structure counterpart. In this way, higher-level structural information can be better preserved than the *value approximation* approaches.

Furthermore, relying on the proposed structured model, we are able to design flexible binary structures according to different tasks and exploit task-specific information or structures to compensate the quantization loss and facilitate training. For example, when transferring Group-Net from image classification to semantic segmentation, we are motivated by the structure of Atrous Spatial Pyramid Pooling (ASPP) [5]. In DeepLab v3 [6] and v3+ [7], ASPP is merely applied on the top of extracted features while each block in the backbone network can employ one atrous rate only. In contrast, we propose to directly apply different atrous rates on parallel binary bases in the backbone network, which is equivalent to absorbing ASPP into the feature extraction stage. Thus, we significantly boost the performance on semantic segmentation, without increasing the computation complexity of the binary convolutions.

In general, it is nontrivial to extend previous *value approximation* based quantization approaches to more challenging tasks such as semantic segmentation (or other general computer vision tasks). However, as will be shown, our Group-Net can be easily extended to other tasks. Nevertheless, it is worth mentioning that value and structure approximation are complementary rather than contradictory. In other words, both are important and should be exploited to obtain highly accurate BNNs.

Our methods are also motivated by those energy-efficient architecture design approaches [8, 21, 23, 49] which seek to replace the traditional expensive convolution with computational efficient convolutional operations (*i.e.*, depthwise separable convolution, 1×1 convolution). Nevertheless, we propose to design binary network architectures for dedicated hardware from the quantization view. We highlight that while most existing quantization works focus on directly quantizing the full-precision architecture, at this point in time we do begin to explore alternative architectures that shall be better suited for dealing with binary weights and activations. In particular, apart from decomposing each group into several binary bases, we also propose to learn the connections between each group by introducing a fusion gate. Moreover, Group-Net can be possibly further improved with Neural Architecture Search methods [37, 55, 56] .

Our contributions are summarized as follows:

- We propose to design accurate BNNs structures from the *structure approximation* perspective. Specifically, we divide the networks into groups and approximate each group using a set of binary bases. We also propose to automatically learn the decomposition by introducing soft connections.
- The proposed Group-Net has strong flexibility and can be easily extended to other tasks. For instance, in this paper we propose Binary Parallel Atrous Convolution (BPAC), which embeds rich multi-scale context into BNNs for accurate semantic segmentation. Group-Net with BPAC significantly improves the performance while maintaining the complexity compared to employ Group-Net only.
- We evaluate our models on ImageNet and PASCAL VOC datasets based on ResNet. Extensive experiments show the proposed Group-Net achieves the state-of-the-art trade-off between accuracy and computational complexity.

We review some relevant works in the sequel.

Network quantization: The recent increasing demand for implementing fixed point deep neural networks on embedded devices motivates the study of network quantization. Fixed-point approaches that use low-bitwidth discrete values to approximate real ones have been extensively explored in the literature [4, 22, 29, 39, 51, 53]. BNNs [22, 39] propose to constrain both weights and activations to binary values (*i.e.*, $+1$ and -1), where the multiply-accumulations can be replaced by purely $\mathrm{xnor}(\cdot)$ and $\mathrm{popcount}(\cdot)$ operations. To make a trade-off between accuracy and complexity, [13, 15, 27, 47] propose to recursively perform residual quantization and yield a series of binary tensors with decreasing magnitude scales. However, multiple binarizations are sequential process which cannot be paralleled. In [29], Lin *et al.* propose to use a linear combination of binary bases to approximate the floating point tensor during forward propagation. This inspires aspects of our approach, but unlike all of these local tensor approximations, we additionally directly design BNNs from a structure approximation perspective.

Efficient architecture design: There has been a rising interest in designing efficient architecture in the recent literature. Efficient model designs like GoogLeNet [45] and SqueezeNet [23] propose to replace 3×3 convolutional kernels with 1×1 size to reduce the complexity while increasing the depth and accuracy. Additionally, separable convolutions are also proved to be effective in Inception approaches [44, 46]. This idea is further generalized as depthwise separable convolutions by Xception [8], MobileNet [21, 43] and ShuffleNet [49] to generate energy-efficient network structure. To avoid handcrafted heuristics, neural architecture search [30, 31, 37, 55, 56] has been explored for automatic model design.

2. Method

Most previous literature has focused on value approximation by designing accurate binarization functions for

weights and activations (*e.g.*, multiple binarizations [13, 15, 27, 29, 47]). In this paper, we seek to binarize both weights and activations of CNNs from a "structure approximation" view. In the following, we first give the problem definition and some basics about binarization in Sec. 2.1. Then, in Sec. 2.2, we explain our binary architecture design strategy. Finally, in Sec. 2.3, we describe how to utilize task-specific attributes to generalize our approach to semantic segmentation.

2.1. Problem definition

For a convolutional layer, we define the input $\mathbf{x} \in \mathbb{R}^{c_{in} \times w_{in} \times h_{in}}$, weight filter $\mathbf{w} \in \mathbb{R}^{c \times w \times h}$ and the output $\mathbf{y} \in \mathbb{R}^{c_{out} \times w_{out} \times h_{out}}$, respectively.

Binarization of weights: Following [39], we approximate the floating-point weight $\mathbf{w}$ by a binary weight filter $\mathbf{b}^w$ and a scaling factor $\alpha \in \mathbb{R}^+$ such that $\mathbf{w} \approx \alpha \mathbf{b}^w$, where $\mathbf{b}^w$ is the sign of $\mathbf{w}$ and α calculates the mean of absolute values of $\mathbf{w}$. In general, $\mathrm{sign}(\cdot)$ is non-differentiable and so we adopt the straight-through estimator [1] (STE) to approximate the gradient calculation. Formally, the forward and backward processes can be given as follows:

$$\begin{aligned} &\text{Forward}: \mathbf{b}^w = \mathrm{sign}(\mathbf{w}), \\ &\text{Backward}: \frac{\partial \ell}{\partial \mathbf{w}} = \frac{\partial \ell}{\partial \mathbf{b}^w} \cdot \frac{\partial \mathbf{b}^w}{\partial \mathbf{w}} \approx \frac{\partial \ell}{\partial \mathbf{b}^w}, \end{aligned} \tag{1}$$

where ℓ is the loss.

Binarization of activations: For activation binarization, we utilize the piecewise polynomial function to approximate the sign function as in [33]. The forward and backward can be written as:

$$\begin{aligned} &\text{Forward}: b^a = \mathrm{sign}(\mathrm{x}), \\ &\text{Backward}: \tfrac{\partial \ell}{\partial x} = \tfrac{\partial \ell}{\partial b^a} \cdot \tfrac{\partial b^a}{\partial x}, \\ &\text{where} \quad \tfrac{\partial b^a}{\partial x} = \begin{cases} 2 + 2x : -1 \le x < 0 \\ 2 - 2x : 0 \le x < 1 \\ 0 : \text{otherwise} \end{cases}. \end{aligned} \tag{2}$$

2.2. Structured Binary Network Decomposition

In this paper, we seek to design a new structural representation of a network for quantization. First of all, note that a float number in computer is represented by a fixed-number of binary digits. Motivated by this, rather than directly doing the quantization via "value decomposition", we propose to decompose a network into binary structures while preserving its representability.

Specifically, given a floating-point residual network Φ with N blocks, we decompose Φ into P binary fragments $[\mathcal{F}_1, ..., \mathcal{F}_P]$, where $\mathcal{F}_i(\cdot)$ can be any binary structure. Note that each $\mathcal{F}_i(\cdot)$ can be different. A natural question arises: can we find some simple methods to decompose the network with binary structures so that the representability can be exactly preserved? To answer this question, we here explore two kinds of architectures for $\mathcal{F}(\cdot)$, namely layer-wise decomposition and group-wise decomposition in Sec. 2.2.1 and Sec. 2.2.2, respectively. After that, we will present a novel strategy for automatic decomposition in Sec. 2.2.3.

2.2.1 Layer-wise binary decomposition

The key challenge of binary decomposition is how to reconstruct or approximate the floating-point structure. The simplest way is to approximate in a layer-wise manner. Let $B(\cdot)$ be a binary convolutional layer and $\mathbf{b}_i^w$ be the binarized weights for the i-th base. In Fig. 1 (c), we illustrate the layer-wise feature reconstruction for a single block. Specifically, for each layer, we aim to fit the full-precision structure using a set of binarized homogeneous branches $\mathcal{F}(\cdot)$ given a floating-point input tensor $\mathbf{x}$:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{K} \lambda_i B_i(\mathbf{x}) = \sum_{i=1}^{K} \lambda_i (\mathbf{b}_i^w \oplus \mathrm{sign}(\mathbf{x})), \tag{3}$$

where $\oplus$ is bitwise operations $\mathrm{xnor}(\cdot)$ and $\mathrm{popcount}(\cdot)$, K is the number of branches and λ_i is the combination coefficient to be determined. During the training, the structure is fixed and each binary convolutional kernel $\mathbf{b}_i^w$ as well as λ_i are directly updated with end-to-end optimization. The scale scalar can be absorbed into batch normalization when doing inference. Note that all B_i's in Eq. (3) have the same topology as the original floating-point counterpart. Each binary branch gives a rough approximation and all the approximations are aggregated to achieve more accurate reconstruction to the original full precision convolutional layer. Note that when $K = 1$, it corresponds to directly binarize the floating-point convolutional layer (Fig. 1 (b)). However, with more branches (a larger K), we are expected to achieve more accurate approximation with more complex transformations.

During the inference, the homogeneous K bases can be parallelizable and thus the structure is hardware friendly. This will bring significant gain in speed-up of the inference. Specifically, the bitwise XNOR operation and bit-counting can be performed in a parallel of 64 by the current generation of CPUs [33, 39]. And we just need to calculate K binary convolutions and K full-precision additions. As a result, the speed-up ratio σ for a convolutional layer can be calculated as:

$$\begin{aligned} \sigma &= \frac{c_{in} c_{out} w h w_{in} h_{in}}{\frac{1}{64}(K c_{in} c_{out} w h w_{in} h_{in}) + K c_{out} w_{out} h_{out}}, \\ &= \frac{64}{K} \cdot \frac{c_{in} w h w_{in} h_{in}}{c_{in} w h w_{in} h_{in} + 64 w_{out} h_{out}}. \end{aligned} \tag{4}$$

We take one layer in ResNet for example. If we set $c_{in} = 256$, $w \times h = 3 \times 3$, $\mathrm{w}_{in} = h_{in} = w_{out} = h_{out} = 28$, $K = 5$, then it can reach $12.45\times$ speedup. But in practice, each branch can be implemented in parallel. And the actual speedup ratio is also influenced by the process of memory read and thread communication.

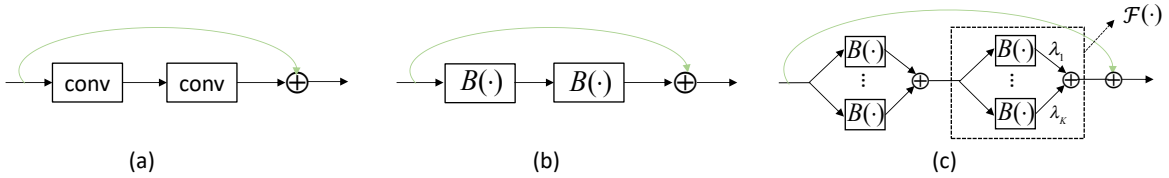

Figure 1: Overview of the baseline binarization method and the proposed layer-wise binary decomposition. We take one residual block with two convolutional layers for illustration. For convenience, we omit batch normalization and nonlinearities. (a): The floating-point residual block. (b): Direct binarization of a full-precision block. (c): Layer-wise binary decomposition in Eq. (3), where we use a set of binary convolutional layers $B(\cdot)$ to approximate a floating-point convolutional layer.

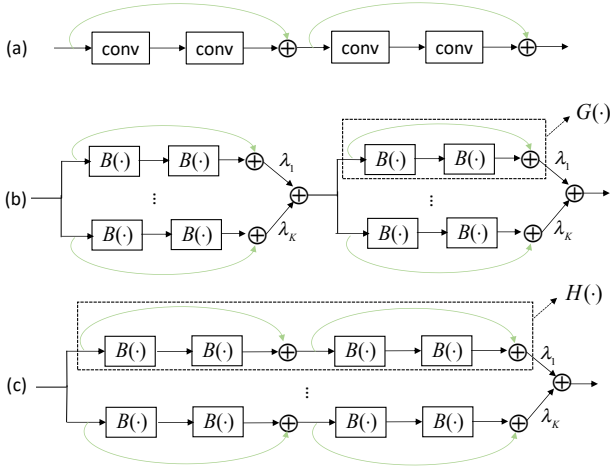

Figure 2: Illustration of the proposed group-wise binary decomposition strategy. We take two residual blocks for description. (a): The floating-point residual blocks. (b): Basic group-wise binary decomposition in Eq. (5), where we approximate a whole block with a linear combination of binary blocks $G(\cdot)$. (c): We approximate a whole group with homogeneous binary bases $H(\cdot)$, where each group consists of several blocks. This corresponds to Eq. (6).

2.2.2 Group-wise binary decomposition

In the layer-wise approach, we approximate each layer with multiple branches of binary layers. Note each branch will introduce a certain amount of error and the error may accumulate due to the aggregation of multi-branches. As a result, this strategy may incur severe quantization errors and bring large deviation for gradients during backpropagation. To alleviate the above issue, we further propose a more flexible decomposition strategy called group-wise binary decomposition, to preserve more structural information during approximation.

To explore the group-structure decomposition, we first consider a simple case where each group consists of only one block. Then, the layer-wise approximation strategy can be easily extended to the group-wise case. As shown in Fig. 2 (b), similar to the layer-wise case, each floating-point group is decomposed into multiple binary groups. However, each group $G_i(\cdot)$ is a binary block which consists of several binary convolutions and floating-point element-wise operations (*i.e.*, ReLU, AddTensor). For example, we can set $G_i(\cdot)$ as the basic residual block [17] which is shown in Fig. 2 (a). Considering the residual architecture, we can decompose $\mathcal{F}(\mathbf{x})$ by extending Eq. (3) as:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{K} \lambda_i G_i(\mathbf{x}), \tag{5}$$

where λ_i is the combination coefficient to be learned. In Eq. (5), we use a linear combination of homogeneous binary bases to approximate one group, where each base G_i is a binarized block. In this way, we effectively keep the original residual structure in each base to preserve the network capacity. As shown in Sec. 4.3.1, the group-wise decomposition strategy performs much better than the simple layer-wise approximation.

Furthermore, the group-wise approximation is flexible. We now analyze the case where each group may contain different number of blocks. Suppose we partition the network into P groups and it follows a simple rule that each group must include one or multiple complete residual building blocks. For the p-th group, we consider the blocks set $T \in \{T_{p-1}+1, ..., T_p\}$, where the index $T_{p-1} = 0$ if $p = 1$. And we can extend Eq. (5) into multiple blocks format:

$$\begin{aligned} \mathcal{F}(\mathbf{x}_{T_{p-1}+1}) &= \sum_{i=1}^{K} \lambda_i H_i(\mathbf{x}), \\ &= \sum_{i=1}^{K} \lambda_i G_i^{T_p}(G_i^{T_p-1}(...(G_i^{T_{p-1}+1}(\mathbf{x}_{T_{p-1}+1}))...)), \end{aligned} \tag{6}$$

where $H(\cdot)$ is a cascade of consequent blocks which is shown in Fig. 2 (c). Based on $\mathcal{F}(\cdot)$, we can efficiently construct a network by stacking these groups and each group may consist of one or multiple blocks. Different from Eq. (5), we further expose a new dimension on each base, which is the number of blocks. This greatly increases the structure space and the flexibility of decomposition. We illustrate several possible connections in Sec. S1 in the supplementary file and further describe how to learn the decomposition in Sec. 2.2.3.

2.2.3 Learning for dynamic decomposition

There is a big challenge involved in Eq. (6). Note that the network has N blocks and the possible number of connections is 2^N. Clearly, it is not practical to enumerate all possible structures during the training. Here, we propose to

solve this problem by learning the structures for decomposition dynamically. We introduce in a fusion gate as the soft connection between blocks $G(\cdot)$. To this end, we first define the input of the i-th branch for the n-th block as:

$$\begin{aligned} C_i^n &= \text{sigmoid}(\theta_i^n), \\ \mathbf{x}_i^n &= C_i^n \odot G_i^{n-1}(\mathbf{x}_i^{n-1}) \\ &+ (1 - C_i^n) \odot \sum_{j=1}^{K} \lambda_j G_j^{n-1}(\mathbf{x}_j^{n-1}), \end{aligned} \tag{7}$$

where $\theta \in \mathbb{R}^K$ is a learnable parameter vector, C_i^n is a gate scalar and $\odot$ is the Hadamard product.

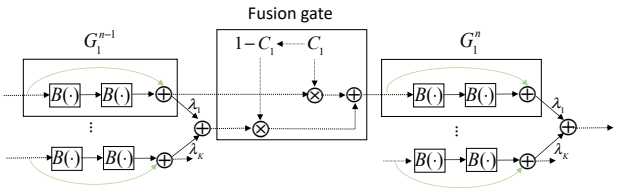

Figure 3: Illustration of the soft connection between two neighbouring blocks. For convenience, we only illustrate the fusion strategy for one branch.

Here, the branch input $\mathbf{x}_i^n$ is a weighted combination of two paths. The first path is the output of the corresponding i-th branch in the $(n-1)$-th block, which is a straight connection. The second path is the aggregation output of the $(n-1)$-th block. The detailed structure is shown in Fig. 3. In this way, we make more information flow into the branch and increase the gradient paths for improving the convergence of BNNs.

Remarks: For the extreme case when $\sum_{i=1}^{K} C_i^n = 0$, Eq. (7) will be reduced to Eq. (5) which means we approximate the $(n-1)$-th and the n-th block independently. When $\sum_{i=1}^{K} C_i^n = K$, Eq. (7) is equivalent to Eq. (6) and we set $H(\cdot)$ to be two consequent blocks and approximate the group as a whole. Interestingly, when $\sum_{n=1}^{N}\sum_{i=1}^{K} C_i^n = NK$, it corresponds to set $H(\cdot)$ in Eq. (6) to be a whole network and directly ensemble K binary models.

2.3. Extension to semantic segmentation

The key message conveyed in the proposed method is that although each binary branch has a limited modeling capability, aggregating them together leads to a powerful model. In this section, we show that this principle can be applied to tasks other than image classification. In particular, we consider semantic segmentation which can be deemed as a dense pixel-wise classification problem. In the state-of-the-art semantic segmentation network, the atrous convolutional layer [6] is an important building block, which performs convolution with a certain dilation rate. To directly apply

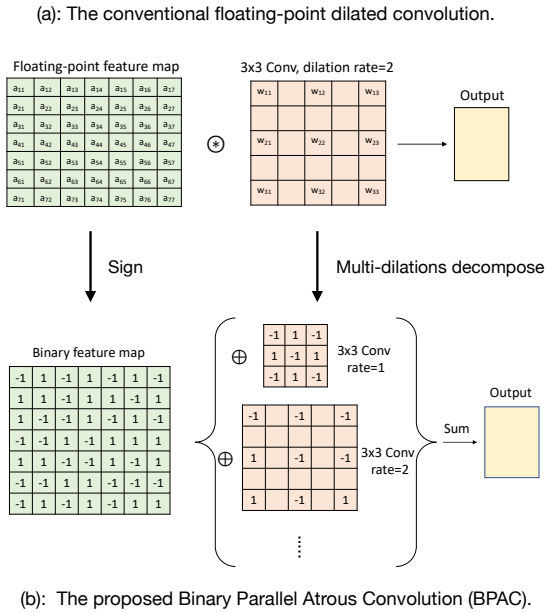

Figure 4: The comparison between conventional dilated convolution and BPAC. For expression convenience, the group only has one convolutional layer. $\circledast$ is the convolution operation and $\oplus$ indicates the XNOR-popcount operations. (a): The original floating-point dilated convolution. (b): We decompose the floating-point atrous convolution into a combination of binary bases, where each base uses a different dilated rate. We sum the output features of each binary branch as the final representation.

the proposed method to such a layer, one can construct multiple binary atrous convolutional branches with the same structure and aggregate results from them. However, we choose not to do this but propose an alternative strategy: we use different dilation rates for each branch. In this way, the model can leverage multiscale information as *a by-product of the network branch decomposition.* It should be noted that this scheme does not incur any additional model parameters and computational complexity compared with the naive binary branch decomposition. The idea is illustrated in Fig. 4 and we call this strategy Binary Parallel Atrous Convolution (**BPAC**).

In this paper, we use the same ResNet backbone in [6, 7] with *output_stride*=8, where the last two blocks employ atrous convolution. In BPAC, we keep $rates = \{2, 3, ..., K, K+1\}$ and $rates = \{6, 7, ..., K+4, K+5\}$ for K bases in the last two blocks, respectively. Intriguingly, as will be shown in Sec. 4.4, our strategy brings so much benefit that using five binary bases with BPAC achieves similar performance as the original full precision network despite the fact that it saves considerable computational cost.

3. Discussions

Complexity analysis: A comprehensive comparison of various quantization approaches over complexity and storage is shown in Table 1. For example, in the previous state-

Table 1: Computational complexity and storage comparison of different quantization approaches. F: full-precision, B: binary, Q_K: K-bit quantization.

Model	Weights	Activations	Operations	Memory saving	Computation Saving
Full-precision DNN	F	F	+, -, $\times$	1	1
[22, 39]	B	B	XNOR-popcount	$\sim 32\times$	$\sim 64\times$
[9, 20]	B	F	+, -	$\sim 32\times$	$\sim 2\times$
[50, 52]	Q_K	F	+, -, $\times$	$\sim \frac{32}{K}\times$	$< 2\times$
[35, 48, 51, 53]	Q_K	Q_K	+, -, $\times$	$\sim \frac{32}{K}\times$	$< \frac{64}{K^2}\times$
[13, 15, 27, 29, 47]	$K \times B$	$K \times B$	+, -, XNOR-popcount	$\sim \frac{32}{K}\times$	$< \frac{64}{K^2}\times$
Group-Net	$K \times (B, B)$		+, -, XNOR-popcount	$\sim \frac{32}{K}\times$	$< \frac{64}{K}\times$

of-the-art binary model ABC-Net [29], each convolutional layer is approximated using K weight bases and K activation bases, which needs to calculate K^2 times binary convolution. In contrast, we just need to approximate several groups with K structural bases. As reported in Sec. 4.2 , we save approximate K times computational complexity while still achieving comparable Top-1 accuracy. Since we use K structural bases, the number of parameters increases by K times in comparison to the full-precision counterpart. But we still save memory bandwidth by $32/K$ times since all the weights are binary in our paper. For our approach, there exists element-wise operations between each group, so the computational complexity saving is slightly less than $\frac{64}{K}\times$.

Differences of Group-net from fixed-point methods: The proposed Group-net with K bases is different from the K-bit fixed-point approaches [35, 48, 51, 53].

We first show how the inner product between fixed-point weights and activations can be computed by bitwise operations. Let a weight vector $\mathbf{w} \in \mathbb{R}^M$ be encoded by a vector $\mathbf{b}_i^w \in \{-1, 1\}^M$, $i = 1, ..., K$. Assume we also quantize activations to K-bit. Similarly, the activations $\mathbf{x}$ can be encoded by $\mathbf{b}_j^a \in \{-1, 1\}^M$, $j = 1, ..., K$. Then, the convolution can be written as

$$Q_K(\mathbf{w}^T)Q_K(\mathbf{x}) = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} 2^{i+j}(\mathbf{b}_i^w \oplus \mathbf{b}_j^a), \quad (8)$$

where $Q_K(\cdot)$ is any quantization function[1].

During the inference, it needs to first get the encoding $\mathbf{b}_j^a$ for each bit by looking up the quantization intervals. Then, it calculates and sums over K^2 times $\text{xnor}(\cdot)$ and $\text{popcount}(\cdot)$. The complexity is about $O(K^2)$. Note that the output range for a single output shall be $[-(2^K-1)^2M, (2^K-1)^2M]$.

In contract, we directly obtain $\mathbf{b}_j^a$ via $\text{sign}(\mathbf{x})$. Moreover, since we just need to calculate K times $\text{xnor}(\cdot)$ and $\text{popcount}(\cdot)$ (see Eq. (3)), and then sum over the outputs, the computational complexity is $O(K)$. For binary convolution, its output range is {-1, 1}. So the value range for each element after summation is $[-KM, KM]$, in which the number of distinct values is much less than that in fixed-point methods.

[1]For simplicity, we only consider uniform quantization in this paper.

In summary, compared with K-bit fixed-point methods, Group-Net with K bases just needs $\sqrt{K}$ computational complexity and saves $(2^K-1)^2/K$ accumulator bandwidth. Even $\sqrt{K}$-bit fixed-point quantization requires more memory bandwidth to feed signal in SRAM or in registers.

Differences of Group-net from multiple binarizations methods: In ABC-Net [29], a linear combination of binary weight/activations bases are obtained from the full-precision weights/activations without being directly learned. In contrast, we directly design the binary network structure, where binary weights are end-to-end optimized. [13, 15, 27, 47] propose to recursively approximate the residual error and obtain a series of binary maps corresponding to different quantization scales. However, it is a sequential process which cannot be paralleled. And all multiple binarizations methods belong to local tensor approximation. In contrast to value approximation, we propose a structure approximation approach to mimic the full-precision network. Moreover, tensor-based methods are tightly designed to local value approximation and are hardly generalized to other tasks accordingly. In addition, our structure decomposition strategy achieves much better performance than tensor-level approximation as shown in Sec. 4.3.1. More discussions are provided in Sec. S2 in the supplementary file.

4. Experiment

We define several methods for comparison as follows: **LBD:** It implements the layer-wise binary decomposition strategy described in Sec. 2.2.1. **Group-Net:** It implements the full model with learnt soft connections described in Sec. 2.2.3. Following Bi-Real Net [32, 33], we apply shortcut bypassing every binary convolution to improve the convergence. **Group-Net**:** Based on Group-Net, we keep the 1×1 downsampling convolution to full-precision similar to [2, 33].

4.1. Implementation details

As in [4, 39, 51, 53], we quantize the weights and activations of all convolutional layers except that the first and the last layer have full-precision. In all ImageNet experiments, training images are resized to 256×256, and a 224×224 crop is randomly sampled from an image or its horizontal

flip, with the per-pixel mean subtracted. We do not use any further data augmentation in our implementation. We use a simple single-crop testing for standard evaluation. No bias term is utilized. We first pretrain the full-precision model as initialization with $\mathrm{Tanh}(\cdot)$ as nonlinearity and fine-tune the binary counterpart. We use Adam [25] for optimization. For training all binary networks, the mini-batch size and weight decay are set to 256 and 0, respectively. The learning rate starts at 5e-4 and is decayed twice by multiplying 0.1 at the 30th and 40th epoch. We train 50 epochs in total. Following [4, 53], no dropout is used due to binarization itself can be treated as a regularization. We apply layer-reordering to the networks as: Sign $\rightarrow$ Conv $\rightarrow$ ReLU $\rightarrow$ BN. Inserting $\mathrm{ReLU}(\cdot)$ after convolution is important for convergence. Our simulation implementation is based on Pytorch [36].

4.2. Evaluation on ImageNet

The proposed method is evaluated on ImageNet (ILSVRC2012) [42] dataset. ImageNet is a large-scale dataset which has $\sim$1.2M training images from 1K categories and 50K validation images. Several representative networks are tested: ResNet-18 [17], ResNet-34 and ResNet-50. As discussed in Sec. 3, binary approaches and fixed-point approaches differ a lot in computational complexity as well as storage consumption. So we compare the proposed approach with binary neural networks in Table 2 and fixed-point approaches in Table 3, respectively.

4.2.1 Comparison with binary neural networks

Since we employ binary weights and binary activations, we directly compare to the previous state-of-the-art binary approaches, including BNN [22], XNOR-Net [39], Bi-Real Net [33] and ABC-Net [29]. We report the results in Table 2 and summarize the following points. 1): The most comparable baseline for Group-Net is ABC-Net. As discussed in Sec. 3, we save considerable computational complexity while still achieving better performance compared to ABC-Net. In comparison to directly binarizing networks, Group-Net achieves much better performance but needs K times more storage and complexity. However, the K homogeneous bases can be easily parallelized on the real chip. In summary, our approach achieves the best trade-off between computational complexity and prediction accuracy. 2): By comparing Group-Net** (5 bases) and Group-Net (8 bases), we can observe comparable performance. *It justifies keeping* 1×1 *downsampling layers to full-precision is crucial for preserving the performance*. 3): For Bottleneck structure in ResNet-50, we find larger quantization error than the counterparts using basic blocks with 3×3 convolutions in ResNet-18 and ResNet-34. The similar observation is also claimed by [3]. We assume that this is mainly attributable to the 1×1 convolutions in Bottleneck. The reason is 1×1 filters are limited to two states only (either 1 or -1) and they have very limited learning power. What's more, the bottleneck structure reduces the number of filters significantly, which means the gradient paths are greatly reduced. In other words, it blocks the gradient flow through BNNs. *Even though the bottleneck structure can benefit full-precision training, it is really needed to be redesigned in BNNs. To increase gradient paths, the* 1×1 *convolutions should be removed.*

4.2.2 Comparison with fix-point approaches

Since we use K binary group bases, we compare our approach with at least $\sqrt{K}$-bit fix-point approaches. In Table 3, we compare our approach with the state-of-the-art fixed-point approaches DoReFa-Net [51], SYQ [12] and LQ-Nets [48]. As described in Sec. 3, K binarizations are more superior than the $\sqrt{K}$-bit width quantization with respect to the resource consumption. Here, we set K=4. DOREFA-Net and LQ-Nets use 2-bit weights and 2-bit activations. SYQ employs binary weights and 8-bit activations. All the comparison results are directly cited from the corresponding papers. LQ-Nets is the current best-performing fixed-point approach and its activations have a long-tail distribution. We can observe that Group-Net requires less memory bandwidth while still achieving comparable accuracy with LQ-Nets.

4.3. Ablation study

Due to the limited space, we provide more experiments in Sec. S1 in the supplementary material.

4.3.1 Layer-wise vs. group-wise binary decomposition

We explore the difference between layer-wise and group-wise design strategies in Table 4. By comparing the results, we find Group-Net outperforms LBD by 7.2% on the Top-1 accuracy. Note that LBD approach can be treated as a kind of *tensor approximation* which has similarities with multiple binarizations methods in [13, 15, 27, 29, 47] and the differences are described in Sec. 3. It strongly shows the necessity for employing the group-wise decomposition strategy to get promising results. We speculate that this significant gain is partly due to the preserved block structure in binary bases. It also proves that apart from designing accurate binarization function, it is also essential to design appropriate structure for BNNs.

4.4. Evaluation on PASCAL VOC

We evaluate the proposed methods on the PASCAL VOC 2012 semantic segmentation benchmark [11] which contains 20 foreground object classes and one background class. The original dataset contains 1,464 (*train*), 1,449 (*val*) and 1,456 (*test*) images. The dataset is augmented by the extra annotations from [16], resulting in 10,582 training images. The performance is measured in terms of averaged

Table 2: Comparison with the state-of-the-art binary models using ResNet-18, ResNet-34 and ResNet-50 on ImageNet. All the comparing results are directly cited from the original papers. The metrics are Top-1 and Top-5 accuracy.

Model		Full	BNN	XNOR	Bi-Real Net	ABC-Net (25 bases)	Group-Net (5 bases)	Group-Net** (5 bases)	Group-Net (8 bases)
ResNet-18	Top-1 %	69.7	42.2	51.2	56.4	65.0	64.8	67.0	**67.5**
	Top-5 %	89.4	67.1	73.2	79.5	85.9	85.7	87.5	**88.0**
ResNet-34	Top-1 %	73.2	-	-	62.2	68.4	68.5	70.5	**71.8**
	Top-5 %	91.4	-	-	83.9	88.2	88.0	89.3	**90.4**
ResNet-50	Top-1 %	76.0	-	-	-	70.1	69.5	71.2	**72.8**
	Top-5 %	92.9	-	-	-	89.7	88.2	90.0	**90.5**

Table 3: Comparison with the state-of-the-art fixed-point models with ResNet-18 on ImageNet. The metrics are Top-1 and Top-5 accuracy.

Model	W	A	Top-1 (%)	Top-5 (%)
Full-precision	32	32	69.7	89.4
Group-Net** (4 bases)	1	1	**66.3**	**86.6**
Group-Net (4 bases)	1	1	64.2	85.6
LQ-Net [48]	2	2	64.9	85.9
DOREFA-Net [51]	2	2	62.6	84.4
SYQ [12]	1	8	62.9	84.6

Table 4: Comparison with Group-Net and LBD using ResNet-18 on ImageNet. The metrics are Top-1 and Top-5 accuracy.

Model	Bases	Top-1 %	Top-5 %
Full-precision	1	69.7	89.4
Group-Net	5	**64.8**	**85.7**
LBD	5	57.6	79.7

Table 5: Performance on PASCAL VOC 2012 validation set.

Model		mIOU	Δ
ResNet-18, FCN-32s	Full-precision	64.9	-
	LQ-Net (3-bit)	62.5	2.4
	Group-Net	60.5	4.4
	Group-Net + BPAC	63.8	1.1
	Group-Net** + BPAC	**65.1**	**-0.2**
ResNet-18, FCN-16s	Full-precision	67.3	-
	LQ-Net (3-bit)	65.1	2.2
	Group-Net	62.7	4.6
	Group-Net + BPAC	66.3	1.0
	Group-Net** + BPAC	**67.7**	**-0.4**
ResNet-34, FCN-32s	Full-precision	72.7	-
	LQ-Net (3-bit)	70.4	2.3
	Group-Net	68.2	4.5
	Group-Net + BPAC	71.2	1.5
	Group-Net** + BPAC	**72.8**	**-0.1**
ResNet-50, FCN-32s	Full-precision	73.1	-
	LQ-Net (3-bit)	70.7	2.4
	Group-Net	67.2	5.9
	Group-Net + BPAC	70.4	2.7
	Group-Net** + BPAC	**71.0**	**2.1**

pixel intersection-over-union (mIOU) over 21 classes. Our experiments are based on the original FCN [34]. For both FCN-32s and FCN-16s, we adjust the dilation rates of the last 2 blocks in ResNet with atrous convolution to make the output stride equal to 8. We first pretrain the binary backbone network on ImageNet dataset and fine-tune it on PASCAL VOC. During fine-tuning, we use Adam with initial learning rate=1e-4, weight decay=1e-5 and batch size=16. We set the number of bases $K = 5$ in experiments. We train 40 epochs in total and decay the learning rate by a factor of 10 at 20 and 30 epochs. We do not add any auxiliary loss and ASPP. We empirically observe full-precision FCN under dilation rates (4, 8) in last two blocks achieves the best performance. The main results are in Table 5.

From the results, we can observe that when all bases using the same dilation rates, there is a large performance gap with the full-precision counterpart. This performance drop **is consistent with** the classification results on ImageNet dataset in Table 2. It proves that the quality of extracted features have a great impact on the segmentation performance. What's more, by utilizing task-specific BPAC, we find significant performance increase with no computational complexity added, which strongly justifies the flexibility of Group-Net. Moreover, we also quantize the backbone network using fixed-point LQ-Nets with 3-bit weights and 3-bit activations. Compared with LQ-Nets, we can achieve comparable performance while saving considerable complexity. In addition, we can observe Group-Net + BPAC based on ResNet-34 even outperform the counterpart on ResNet-50. This shows the widely used bottleneck structure is not suited to BNNs as explained in Sec. 4.2.1. We provide more analysis in Sec. S3 in the supplementary file.

5. Conclusion

In this paper, we have begun to explore highly efficient and accurate CNN architectures with binary weights and activations. Specifically, we have proposed to directly decompose the full-precision network into multiple groups and each group is approximated using a set of binary bases which can be optimized in an end-to-end manner. We also propose to learn the decomposition automatically. Experimental results have proved the effectiveness of the proposed approach on the ImageNet classification task. Moreover, we have generalized Group-Net from image classification task to semantic segmentation and achieved promising performance on PASCAL VOC. We have implemented the homogeneous multi-branch structure on CPU and achieved promising acceleration on test-time inference.

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

References

[1] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

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

[3] J. Bethge, H. Yang, C. Bartz, and C. Meinel. Learning to train a binary neural network. *arXiv preprint arXiv:1809.10463*, 2018. 7

[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5918–5926, 2017. 1, 2, 6, 7

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 2

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 5

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *Proc. Eur. Conf. Comp. Vis.*, 2018. 1, 2, 5

[8] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1251–1258, 2017. 1, 2

[9] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 3123–3131, 2015. 6

[10] A. Ehliar. Area efficient floating-point adder and multiplier with ieee-754 compatible semantics. In *Field-Programmable Technology (FPT), 2014 International Conference on*, pages 131–138. IEEE. 1

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comp. Vis.*, 88(2):303–338, 2010. 7

[12] J. Faraone, N. Fraser, M. Blott, and P. H. Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 7, 8

[13] J. Fromm, S. Patel, and M. Philipose. Heterogeneous bitwidth binarization in convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 2018. 2, 3, 6, 7

[14] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna. Analysis of high-performance floating-point arithmetic on fpgas. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 149. IEEE, 2004. 1

[15] Y. Guo, A. Yao, H. Zhao, and Y. Chen. Network sketching: Exploiting binary structure in deep cnns. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5955–5963, 2017. 2, 3, 6, 7

[16] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proc. Eur. Conf. Comp. Vis.*, 2011. 7

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. 1, 4, 7

[18] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, volume 2, page 6, 2017. 1

[19] L. Hou and J. T. Kwok. Loss-aware weight quantization of deep networks. In *Proc. Int. Conf. Learn. Repren.*, 2018. 1

[20] L. Hou, Q. Yao, and J. T. Kwok. Loss-aware binarization of deep networks. In *Proc. Int. Conf. Learn. Repren.*, 2017. 1, 6

[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[22] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 4107–4115, 2016. 1, 2, 6, 7

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2

[24] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 1

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Repren.*, 2015. 7

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1097–1105, 2012. 1

[27] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao. Performance guaranteed network acceleration via high-order residual quantization. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2584–2592, 2017. 2, 3, 6, 7

[28] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 1

[29] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 344–352, 2017. 2, 3, 6, 7

[30] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 2

[31] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *Proc. Int. Conf. Learn. Repren.*, 2018. 2

[32] Z. Liu, W. Luo, B. Wu, X. Yang, W. Liu, and K.-T. Cheng. Bi-real net: Binarizing deep network towards real-network performance. *arXiv preprint arXiv:1811.01335*, 2018. 6

[33] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 3, 6, 7

[34] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015. 1, 8

[35] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *Proc. Int. Conf. Learn. Repren.*, 2018. 1, 6

[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Proc. Adv. Neural Inf. Process. Syst. Workshops*, 2017. 7

[37] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *Proc. Int. Conf. Mach. Learn.*, 2018. 2

[38] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. In *Proc. Int. Conf. Learn. Repren.*, 2018. 1

[39] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 525–542, 2016. 1, 2, 3, 6, 7

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 779–788, 2016. 1

[41] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 91–99, 2015. 1

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vis.*, 115(3):211–252, 2015. 7

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4510–4520, 2018. 2

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. AAAI Conf. on Arti. Intel.*, volume 4, page 12, 2017. 2

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1–9, 2015. 2

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2818–2826, 2016. 2

[47] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *Proc. AAAI Conf. on Arti. Intel.*, pages 2625–2631, 2017. 2, 3, 6, 7

[48] D. Zhang, J. Yang, D. Ye, and G. Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 1, 6, 7, 8

[49] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 2

[50] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *Proc. Int. Conf. Learn. Repren.*, 2017. 6

[51] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1, 2, 6, 7, 8

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

[53] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Towards effective low-bitwidth convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 1, 2, 6, 7

[54] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 883–894, 2018. 1

[55] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[56] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 2