Structured Binary Neural Networks for Image Recognition

Bohan Zhuang, Chunhua Shen, Mingkui Tan, Peng Chen, Lingqiao Liu, and Ian Reid

Abstract—In this paper, we propose to train binarized convolutional neural networks (CNNs) that are highly desired in mobile devices with limited power capacity and computing resources. Previous works on quantizing CNNs often seek to approximate the floating-point information of weights and/or activations using a set of discrete values. Such methods, which we call value approximation, typically are built on the same architecture to the full-precision networks. Instead, we take a novel "structure approximation" view for network quantization—it is possible yet valuable to exploit flexible architecture transformation when learning low-bit networks, which can achieve even better performance than the original networks. In particular, we propose a "group decomposition" strategy for networks, called Group-Net, which divides a network into desired groups. Interestingly, via our Group-Net strategy, each full-precision group can be effectively reconstructed by aggregating a set of homogeneous binary branches. We also propose to learn effective connections among groups to improve the representation capability. More importantly, the proposed Group-Net shows strong flexibility to other tasks. For instance, we extend the Group-Net for accurate semantic segmentation by embedding rich context into the binary structure. The proposed Group-Net also shows strong power for accurate object detection. Experiments on image classification, semantic segmentation and object detection tasks demonstrate the superior performance of the proposed methods over various quantized networks in the literature. Moreover, the speedup evaluation comparing with related quantization strategies is analyzed on GPU platforms, which serves as a strong benchmark for further research.

Index Terms—Binary neural networks, quantization, image classification, semantic segmentation, object detection.

1 Introduction

Deep convolutional neural networks have achieved significant breakthroughs in many machine learning tasks, such as image classification [1], [2], object segmentation [3], [4] and object detection [5], [6]. However, deep models often require billions of FLOPs for inference, which makes them infeasible for many real-time applications especially on resource constrained mobile platforms. To solve this, many existing works focus on network pruning [7], [8], [9], low-bit quantization [10], [11] and/or efficient architecture design [12], [13]. Among them, the quantization approaches seek to represent the weights and/or activations with low bitwidth fixed-point integers. In this way, the dot product can be computed by several XNOR-popcount bitwise operations. Binarization [14], [15], which is an extreme quantization approach, seeks to represent the weights and activations by a single bit (e.g., $+1$ or -1). Binarization has gained great attention recently since the XNOR of two bits requires only a single logic gate instead of using hundreds units for floating point multiplications [16], [17]. 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, including binarization, seeks to quantize weights and/or activations by preserving most of the representational ability of the original network. In this sense, we call them *value approximation* methods, which can be mainly divided into two categories. The first category seeks to design more effective optimization algorithms to find better local minima for quantized weights. These works either introduce knowledge distillation [10], [18], [19] or use loss-aware objectives [20], [21]. The second category approaches focus on improving the quantization function [22], [23], [24], by learning suitable mappings between discrete values and their floating-point counterparts.

However, the value approximation based approaches have a natural limitation that it is merely a local approximation to the original network. Moreover, designing a good quantization function is highly non-trivial especially for BNNs, since the quantization process essentially is non-differentiable and the gradients can only be roughly approximated. Last, these methods often lack of adaptive ability to general tasks. For example, these methods may work well on image classification tasks, but may not achieve promising quantization performance on segmentation and detection tasks.

In this paper, we investigate the task of quantization from a new perspective called *structure approximation*. We find that, instead of directly approximating the original network, it is possible yet valuable to redesign a binary architecture that can directly match the capability of the floating-point model. In particular, we propose a "group decomposition" strategy called Group-Net which seeks to partition a full-precision model into groups. One of the key points is, based on the proposed Group-Net strategy, we are able to use a set of binary bases to approximate its floating-point structure counterpart. This kind of design brings two benefits.

On one hand, Group-Net enables more flexible trade-off between computational complexity and accuracy. Specif-

- *B. Zhuang is with Monash University, Australia. C. Shen, P. Chen, L. Liu and I. Reid are with The University of Adelaide, Australia. E-mail: (firstname.lastname@adelaide.edu.au). C. Shen is also with Monash University. This work was done when B. Zhuang was with The University of Adelaide. C. Shen is the corresponding author.*
- *M. Tan is with South China University of Technology, China. E-mail: (mingkuitan@scut.edu.cn).*

ically, Group-Net enables fine-grained quantization levels that can be any positive integers (except 1) while fixed-point methods [10], [22] require the quantization levels to be exponential power of 2. As a result, Group-Net can achieve the fine-grained bit-width by directly controlling the number of bases, which is more optimal for balancing the efficiency and accuracy of the overall network.

On the other hand, the higher-level structural information can be better preserved than the *value approximation* approaches. In practice, while the *value approximation* based approaches show promising performance on image classification tasks, they often performs poorly on more challenging tasks such as semantic segmentation and object detection. Relying on the proposed structured strategy, we are able to exploit task-specific information or structures and further design flexible binary structures according to specific tasks to compensate the quantization loss for general tasks.

First, to deal with the semantic segmentation task, we are motivated by Atrous Spatial Pyramid Pooling (ASPP) [4], [25], which is built on top of extracted features of the backbone network. To capture the multi-scale context, 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. As will be shown, our strategy significantly boosts the performance on semantic segmentation without increasing the computational complexity of the binary convolutions. Second, we further extend the proposed approach to building quantized networks for object detection. Building low-precision networks for object detection is more challenging since detection needs the network outputs richer information, such as locations of bounding boxes. There has been several works in literature to address the quantized object detectors [11], [26], [27]. However, there still exists a significant performance drop of lower-precision quantized detectors comparing to their full-precision counterpart. To tackle this problem, we apply our Group-Net and propose a new design modification to better accommodate the quantized object detector. Last but not least, it is worth mentioning that our structure approximation strategy and the value approximation strategy are complementary rather than contradictory. In fact, both are important and should be exploited to obtain highly accurate BNNs.

Our methods are closely related to those energy-efficient architecture design approaches [12], [13], [28], [29], which seek to replace the traditional expensive convolution with computational efficient convolutional operations (e.g., depthwise separable convolution). Nevertheless, we propose in this paper to redesign binary network architectures 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 potentially further improved with Neural Architecture Search methods [30], [31], [32] .

Our contributions are summarized as follows:

- We propose to design accurate BNNs structures from the *structure approximation* perspective. Specifically, we divide the network 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 tasks other than image classification. 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.
- To the best of our knowledge, we may be among the pioneering approaches to apply binary neural networks on general semantic segmentation and object detection tasks.
- We develop the underlying implementation to evaluate the execution speed of Group-Net and make comparison with other bit configurations on various platforms.
- We evaluate our models on ImageNet, PASCAL VOC and COCO datasets based on various architectures. Extensive experiments show the proposed Group-Net achieves the state-of-the-art trade-off between accuracy and computational complexity.

This paper extends our preliminary results, which appeared in [33], in several aspects. 1) We develop the acceleration code on resource constrained platforms and conduct the speedup evaluation comparing with various quantization methods. 2) We make more analysis on differences and advantages of Group-Net over other related quantization strategies. 3) In addition to image classification and semantic segmentation, we further extend Group-Net to object detection. In particular, we propose several modifications to quantized object detection and our Group-Net outperforms the comparison methods. 4) For image classification, we conduct more ablation studies and experiments on more architectures and provide more analysis. 5) For semantic segmentation, we also implement on DeepLabv3 and provide useful instructions. We have also provided more technical details here.

2 Related Work

Network quantization: The recent increasing demand for implementing fixed-point deep neural networks on embedded devices motivates the study of low-bit network quantization. Quantization based methods represent the network weights and/or activations with very low precision, thus yielding highly compact DNN models compared to their floating-point counterparts. BNNs [14], [15] propose to constrain both weights and activations to binary values (i.e., $+1$ and -1), where the multiplication-accumulations can be replaced by purely $\mathrm{xnor}(\cdot)$ and $\mathrm{popcount}(\cdot)$ operations, which are in general much faster. However, BNNs still suffer from significant accuracy decrease compared with the full precision counterparts. To narrow this accuracy gap,

ternary networks [34], [35] and even higher bit fixed-point quantization [22], [36] methods are proposed.

In general, quantization approaches target at tackling two main problems. On one hand, some works target at designing more accurate quantizer to minimize information loss. For the uniform quantizer, works in [37], [38] explicitly parameterize and optimize the upper and/or lower bound of the activation and weights. To reduce the quantization error, non-uniform approaches [24], [39] are proposed to better approximate the data distribution. In particular, LQ-net [24] proposes to jointly optimize the quantizer and the network parameters. On the other hand, because of the non-differentiable quantizer, some literature focuses on relaxing the discrete optimization problem. A typical approach is to train with regularization [40], [41], where the optimization problem becomes continuous while gradually adjusting the data distribution towards quantization levels. Moreover, Hou *et al.* [20], [21] propose the loss-aware quantization by directly optimizing the discrete objective function. Apart from the two challenges, with the popularization of neural architecture search (NAS), Wang *et al.* [42] further propose to employ reinforcement learning to automatically determine the bit-width of each layer without human heuristics.

To well balance accuracy and complexity, several works [43], [44], [45], [46], [47] propose to employ a linear combination of binary tensors to approximate the filters and/or activations while still possessing the advantage of binary operations. In particular, Guo *et al.* [43] recursively performs residual quantization on pretrained full-precision weights and does convolution on each binary weight base. Similarly, Li *et al.* [44] propose to expand the input feature maps into binary bases in the same manner. And Lin *et al.* [45] further expand both weights and activations with a simple linear approach. Unlike the previous local tensor approximation approaches, we directly design BNNs from a structure approximation perspective and show strong generalization on a few mainstream computer vision tasks.

Hardware Implementation: In addition to the quantization algorithms design, the implementation frameworks and acceleration libraries [11], [48], [49], [50], [51] are indispensable to expedite the quantization technique to be deployed on energy-efficient edge devices. For example, TBN [52] focuses on the implementation of ternary activation and binary weight networks. daBNN [53] targets at the inference optimization of BNNs on ARM CPU devices. GXNOR-Net [54] treats TNNs as a kind of sparse BNNs and propose an acceleration solution on dedicated hardware platforms. In this paper, we develop the acceleration code for BNNs, Group-Net and fixed-point quantization on GPU platforms. We also compare the accuracy and efficiency trade-offs between them.

Efficient architecture design: There has been a rising interest in designing efficient architecture recently. Efficient model designs like GoogLeNet [55] and SqueezeNet [28] 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 [56], [57]. This idea is further generalized as depthwise separable convolutions as in Xception [12], MobileNet [13], [58] and ShuffleNet [29] to obtain energy-efficient network structure. To avoid handcrafted heuristics to explore the enormous design space, NAS [30], [31], [32], [59], [60] has been explored for automatically sample the design space and improve the model quality.

Semantic segmentation: Deep learning based semantic segmentation is popularized by the Fully Convolutional Networks (FCNs) [3]. Recent prominent directions have emerged: using the encoder-encoder structure [4], [61]; relying on dilated convolutions to keep the reception fields without downsampling the spatial resolution too much [25], [62]; employing multi-scale feature fusion [63], [64]; employment of probabilistic graphical models [65], [66].

However, these approaches typically focus on designing complex modules for improving accuracy while sacrificing the inference efficiency to some extent. To make semantic segmentation applicable, several methods have been proposed to design real-time semantic segmentation models. Recently, works of [67], [68] apply neural architecture search for exploring more accurate models with less Multi-Adds operations. Yu *et al.* [69] propose BiSeNet, where a spatial path extracts high-resolution features and a context path obtains sufficient receptive fields to achieve high speed and accuracy. ESPNet [62], [70] design efficient spatial pyramid for real-time semantic segmentation under resource constraints. In contrast, we instead propose to accelerate semantic segmentation frameworks from the quantization perspective, which is parallel to the above approaches. Given a pretrained full-precision model, we can replace multiply-accumulations by the XNOR-popcount operations, which would bring great benefits for ARM and FPGA platforms. We may be the first to apply binary neural networks on semantic segmentation and achieve promising results.

Object detection: Object detection has shown great success with deep neural networks. As one of the dominant detection framework, two-stage detection methods [6], [71], [72] first generate region proposals and then refine them by subsequent networks. The popular method Faster-RCNN [6] first proposes an end-to-end detection framework by introducing a region proposal network (RPN). Another main category is the one-stage methods which are represented by YOLO [5], [73], [74] and SSD [75]. The objective is to improve the detection efficiency by directly classifying and regressing the pre-defined anchors without the proposal generation step. RetinaNet [76] proposes a novel focal loss to tackle the extreme foreground-background class imbalance encountered during training in one-stage detectors.

Two recent developing trends in object detection are worth to mention. On the one hand, designing light-weight detection frameworks is crucial since mobile applications usually require real-time, low-power and fully embeddable. For example, the work of [77] and [27] propose to train a tiny model by distilling knowledge from a deeper teacher network. MNasNet [78] propose to automatically search for mobile CNNs which achieve better mAP quality than MobileNets for COCO object detection. On the other hand, anchor-free detection [79], [80] is now gaining more and more attention. Since detection becomes proposal free and anchor free, it can avoid the hyperparameters and complicated computation related to anchor boxes. Moreover, Tian *et al.* [79] propose a simple fully convolutional anchor-free one-stage detector that achieves comparable performance

with the anchor-based one-stage detectors. In this paper, we explore to unify these two cutting-edge trends by proposing to binarize the one-stage anchor-free detection framework. Note that, we are the first to train a binary object detection model in the literature.

3 Proposed Method

Most previous methods in the literature have focused on value approximation by designing accurate binarization functions for weights and activations (e.g., multiple binarizations [43], [44], [45], [46], [47]). In this paper, we seek to binarize both weights and activations of CNNs from a "structure approximation" view. In the following, we first define the research problem and present some basic knowledge about binarization in Section 3.1. Then, in Section 3.2, we explain our binary architecture design strategy. In Section 3.3, we further provide the complexity analysis. Finally, in Section 3.4 and Section 3.5, we describe how to use task-specific attributes to generalize our approach to semantic segmentation and object detection, respectively.

3.1 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 [15], 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 [81] (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 [82]. The forward and backward can be written as:

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

3.2 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 Section 3.2.1 and Section 3.2.2, respectively. Then we present a novel strategy for automatic decomposition in Section 3.2.3.

3.2.1 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 Figure 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 $\mathbf{x}$:

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

where $\oplus$ is bitwise operations $\mathrm{xnor}(\cdot)$ and $\mathrm{popcount}(\cdot)$, K is the number of branches. During the training, the structure is fixed and each binary convolutional kernel $\mathbf{b}_i^w$ is 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 (Figure 1 (b)). However, with more branches (a larger K), we are expected to achieve more accurate approximation with more complex transformations.

Different from [33], we remove the floating-point scales for two reasons. First, the scales can be absorbed into batch normalization layers. Second, we empirically observe that the learnt scales for different branches differ a little and removing them does not influence the performance.

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 [15], [82]. 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:

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

We take one layer in ResNet for example. If we set $c_{in} = 256$, $w \times h = 3 \times 3$, $w_{in} = h_{in} = w_{out} = h_{out} = 28$, $K = 5$, then it can reach 12.45× speedup. But in practice, each branch can be implemented in parallel. And the actual

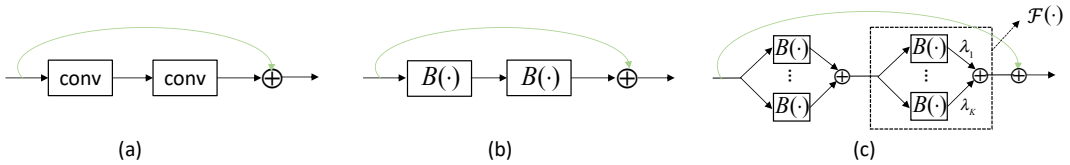


Fig. 1: Overview of the baseline binarization method and the proposed layer-wise binary decomposition. We take one residual block with two convolutional layers for illustration. For convenience, we omit batch normalization and nonlinearities. (a): The floating-point residual block. (b): Direct binarization 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.

speedup ratio is also influenced by the process of memory read and thread communication.

3.2.2 Group-wise binary decomposition

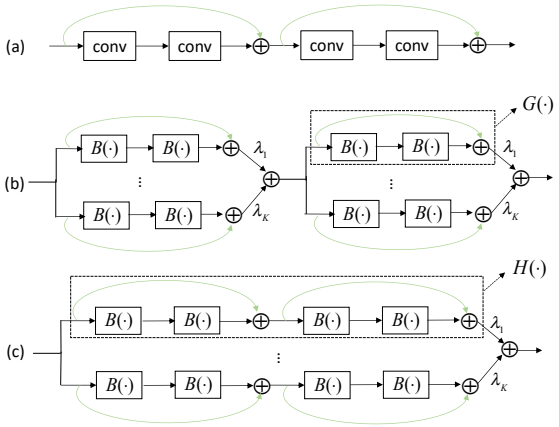


Fig. 2: Illustration of the proposed group-wise binary decomposition strategy. We take two residual blocks for description. (a): The floating-point residual blocks. (b): Basic group-wise binary decomposition in Eq. (5), where we approximate a whole block with a linear combination of binary blocks $G(\cdot)$.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

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 back-propagation. 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 Figure 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 fixed-point operations (i.e., AddTensor). For example, we can set $G_i(\cdot)$ as the basic residual block [2] which is shown in Figure 2 (a). Considering the residual architecture, we can decompose $\mathcal{F}(\mathbf{x})$ by extending Eq. (3) as:

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

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. Thus, we effectively keep the original residual structure in each base to preserve the network capacity. As shown in Section 5.2.1, 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 $p = 1$. We can extend Eq. (5) into multiple blocks format:

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

where $H(\cdot)$ is a cascade of consequent blocks which is shown in Figure 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 Figure 7 and further describe how to learn the decomposition in Section 3.2.3.

3.2.3 Learning for decomposition

There is a significant 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:

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

where $\boldsymbol{\theta} \in \mathbb{R}^K$ is a learnable parameter vector, C_i^n is a gate scalar and $\odot$ is the Hadamard product. And we empirically observe that using a learnable scale θ that shares among branches does not influence the performance and it can

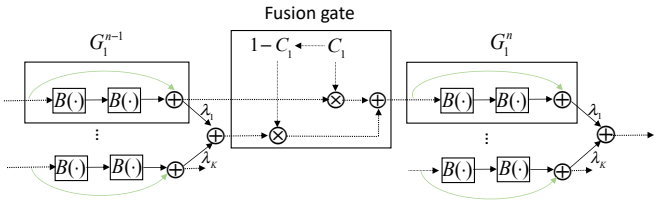
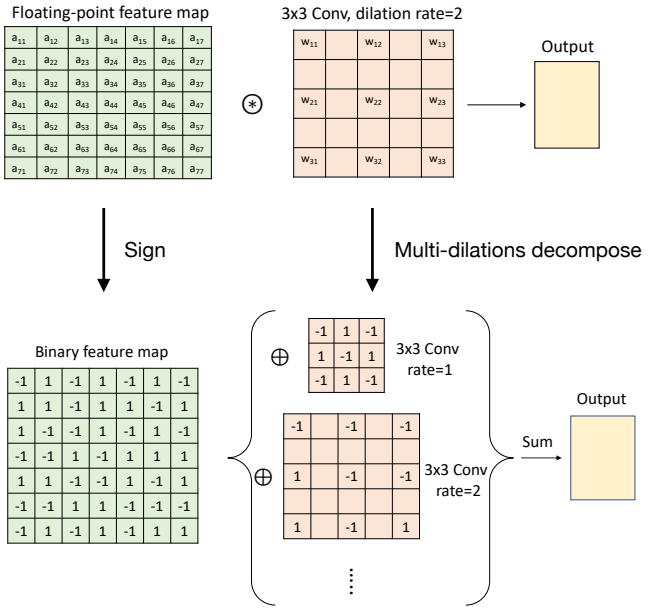


Fig. 3: Illustration of the soft connection between two neighbouring blocks. For convenience, we only illustrate the fusion strategy for one branch.

(a): The conventional floating-point dilated convolution.



(b): The proposed Binary Parallel Atrous Convolution (BPAC).

Fig. 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.

be absorbed into batch normalization (BN) layers during inference.

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 direct connection. The second path is the aggregation output of the $(n-1)$-th block. The detailed structure is shown in Figure 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.

3.3 Complexity analysis

A comprehensive comparison of various quantization approaches over complexity and storage is shown in Table 1. For example, in the previous state-of-the-art binary model ABC-Net [45], 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 Section 5.1 , 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 $\frac{64}{K}\times$.

3.4 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 [25] is an important building block, which performs convolution with a certain dilation rate. To directly apply 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 Figure 4 and we term this strategy Binary Parallel Atrous Convolution (**BPAC**).

In this work, we use the same ResNet backbone in [4], [25] with *output stride=8*, where the last two stages employ atrous convolution. In BPAC, we keep $rates = \{2, ..., K+1\}$ and $rates = \{6, ..., K+5\}$ for K bases in the last two blocks, respectively. Intriguingly, as will be shown in Section 5.3, 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.

3.5 Extension to object detection

We further generalize Group-Net to the object detection task. We work on the latest one-stage anchor-free detector FCOS [79] as shown in Figure 5. FCOS is built on the multi-level prediction framework FPN [84]. Its main difference between anchor-based detectors is that it directly views locations as training samples which is the same as in FCNs for semantic segmentation instead of using anchor boxes in

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
[14], [15]	B	B	XNOR-popcount	$\sim 32\times$	$\sim 64\times$
[21], [83]	B	F	$+, -$	$\sim 32\times$	$\sim 2\times$
[35], [36]	Q_K	F	$+, -, \times$	$\sim \frac{32}{K}\times$	$< 2\times$
[10], [19], [22], [24]	Q_K	Q_K	$+, -, \times$	$\sim \frac{32}{K}\times$	$< \frac{64}{K^2}\times$
[43], [44], [45], [46], [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	$\sim \frac{32}{K}\times$	$< \frac{64}{K}\times$

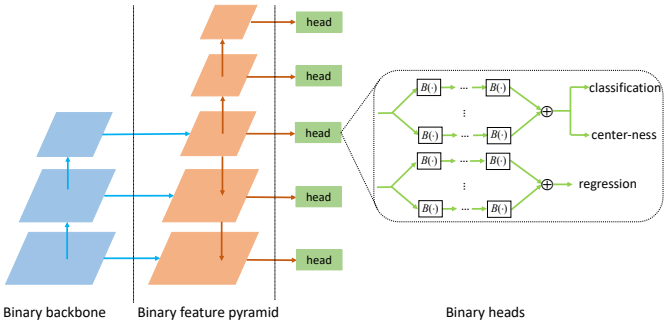


Fig. 5: Illustration of the proposed binary detection framework. We partition the whole framework into three parts, namely, binary backbone, binary feature pyramid and binary heads.

anchor-based detectors. More details can be referred to the original paper [79].

The overall detection framework consists of backbone, feature pyramid and heads. We directly use the backbone network pretrained on the ImageNet classification task to initialize the detection backbone. For feature pyramid structure, it attaches a 1×1 and 3×3 convolutional layer at each resolution to adapt the feature maps. Since it lacks of structural information like the backbone network, we therefore apply the layer-wise binary decomposition in Section 3.2.1. Furthermore, the FPN heads occupy a large portion of complexity in the whole detection framework. And each head is comprised of several consequent layers which is similar to a basic residual block. To preserve the structural information, following the spirit of group-wise binary decomposition strategy in Section 3.2.2, we propose to approximate each head as a whole. The structure is illustrated in Figure 5.

We note that except the last layers for classification, center-ness and regression, other parameters of the heads are not shared across all feature pyramid levels which is the opposite of the full-precision counterpart. The reason can be attributed to the fact that the multi-level semantic information is represented by activation magnitude at different pyramid levels. In the full-precision setting, the shared heads are enough to adapt the different magnitude for classification and regression. However, in the binary setting, the feature magnitude is at the same scale across the pyramid levels since the activations are constrained to $\{-1, 1\}$. With the same input magnitude, the heads should learn independent parameters to capture multi-level information. More optimization details are explained in Section 5.4.

4 Discussions

Differences from fixed-point quantization approaches: Our Group-Net is different from fixed-point quantization approaches [10], [19], [22], [24] in both the quantization algorithm and its underlying inference implementation. Note that in conventional fixed-point methods, the inner product between fixed-point weights and activations can be computed by bitwise operations. Let $\mathbf{w} \in \mathbb{R}^M$ and $\mathbf{a} \in \mathbb{R}^M$ be the weights and activations, respectively. they can be encoded by a linear combination of binary bases, respectively. In particular, $\mathbf{w}$ and $\mathbf{a}$ can be encoded by $\mathbf{b}_i^w \in \{-1, 1\}^M$ and $\mathbf{b}_i^a \in \{-1, 1\}^M$, where $i = 1, ..., P$, respectively. Let $Q_P(\cdot)$ be any quantization function, and for simplicity, we here consider uniform quantization only. Then, the inner product of $\mathbf{w}$ and $\mathbf{a}$ can be approximated by

$$Q_P(\mathbf{w}^{\mathrm{T}})Q_P(\mathbf{a}) = \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} 2^{i+j}(\mathbf{b}_i^w \oplus \mathbf{b}_j^a). \tag{8}$$

where $\oplus$ indicates the binary inner product, which can be efficiently implemented by popcount and xnor bit-wise instructions that are commonly equipped in modern computers. According to Eq. (8), the computational complexity is $O(MP^2)$ for the fixed-point inner product. Bearing the structured binary decomposition in Section 3.2 in mind, it can be easily realized that the proposed Group-Net with K bases has the same computational complexity with the conventional P-bit fixed point quantization when $K = 2^P$. However, compared with the P-bit fixed point quantization, the proposed Group-Net with 2^P bases introduces extra additions from the fusion of decomposition branches. It is worth noting that the high-precision additions can be efficiently merged into the im2col operation of the succeeding layer during implementation. For im2col operation, most of the time is spend on data rearrangement, an extra tensor addition will have a very limited influence on its execution time. Moreover, the extra additions only account for a small portion of the overall network complexity, which intrinsically limits the impact on the overall speed. To justify our analysis, we provide comprehensive evaluations in Section 6.

To be emphasized, the proposed Group-Net enables much more flexible design of the quantization algorithm. Benefit from the structured binary decomposition, the proposed Group-Net allows more fine-grained exploration space, where the quantization levels can be chosen from the continuous positive integer domain. In contrast, conventional fixed-point quantization requires the quantization levels to be the power of 2. For example, the bit-width of Group-Net with 5 bases is between 2-bit and 3-bit. Thus,

Group-Net enjoys a flexible trade-off between complexity and accuracy by setting appropriate K. Moreover, as introduced in Section 3.4 and Section 3.5, our Group-Net is demonstrated to be more efficient in exploiting task-specific information or structures to compensate the quantization information loss.

Based on the above analysis, we summarize that the proposed Group-Net introduces small additional run-time cost caused by the extra addition operations, however, potentially provides benefits in algorithm flexibility and performance.

Differences of Group-Net from mixed-precision quantization: Mixed-precision quantization [42], [85] aims to assign optimal bit-widths for different layers, resulting in a fine-grained bitwidth of the overall network. However, each layer is still quantized in a fixed-point manner. In contrast, Group-Net enables each layer to be quantized in a fine-grained manner by controlling the number of bases, which is more flexible than the mixed-precision counterpart.

Differences of Group-Net from other multiple binarizations methods: In ABC-Net [45], 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. [43], [44], [46], [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 Section 5.2.1.

Relation to ResNeXt [86]: The homogeneous multi-branch architecture design shares the spirit of ResNeXt and enjoys the advantage of introducing a "cardinality" dimension. However, our objectives are totally different. ResNeXt aims to increase the capacity while maintaining the complexity. To achieve this, it first divides the input channels into groups and perform efficient group convolutions implementation. Then all the group outputs are aggregated to approximate the original feature map. In contrast, we first divide the network into groups and directly replicate the floating-point structure for each branch while both weights and activations are binarized. In this way, we can reconstruct the full-precision structure via aggregating a set of low-precision transformations for complexity reduction in the energy-efficient hardware. Furthermore, our structured transformations are not restricted to a single residual block as in ResNeXt.

Strong flexibility of Group-Net: The group-wise approximation approach can be efficiently integrated with Neural Architecture Search (NAS) frameworks [30], [31], [32], [59], [87] to explore the optimal architecture. For instance, based on Group-Net, we can further add the number of bases, depth as well as the resolution into the search space following [88]. The proposed approach can also be combined with knowledge distillation strategy as in [10], [19]. In this way, we expect to further decrease the number of bases while maintaining the performance.

5 Experiments

We define several methods for comparison as follows: **LBD:** It implements the layer-wise binary decomposition strategy described in Section 3.2.1. **Group-Net:** It implements the full model with learnt soft connections described in Section 3.2.3. Following Bi-Real Net [82], [89], we apply skip connection bypassing every binary convolution to improve the convergence. Note that due to the binary convolution, skip connections are high-precision which can be efficiently implemented using fixed-point addition. **Group-Net**:** Based on Group-Net, we keep the 1×1 downsampling skip connections to high-precision (i.e., 8-bit) similar to [82], [90].

5.1 Evaluation on ImageNet

The proposed method is first evaluated on ImageNet (ILSVRC2012) [91] 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 [2], ResNet-34 and ResNet-50. As discussed in Section 4, 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.

5.1.1 *Implementation details*

As in [10], [15], [22], [23], we binarize the weights and activations of all convolutional layers except that the first and the last layer are quantized to 8-bit. In all ImageNet experiments, training images are resized to 256×256, and a 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted. We do not use any further data augmentation in our implementation. We use a simple single-crop testing for standard evaluation. No bias term is utilized. Training is divided into two stages. We first pretrain the full-precision model as initialization and fine-tune the binary counterpart. For pretraining, we use SGD with a initial learning rate of 0.05, a momentum of 0.9 and a weight decay of 1e-4. We use $\mathrm{Tanh}(\cdot)$ as nonlinearity instead of $\mathrm{ReLU}(\cdot)$. For fine-tuning, we use Adam [92] for optimization with a mini-batch size of 256 and a weight decay of 0, respectively. The learning rate starts at 5e-4. For both stages, the learning rate is decayed twice by multiplying 0.1 at the 25th and 35th epoch, and we train a maximum 40 epochs in each stage. Following [10], [23], no dropout is used due to binarization itself can be treated as a regularization. We apply layer-reordering to the networks as: $\mathrm{Sign} \rightarrow \mathrm{Conv} \rightarrow \mathrm{ReLU} \rightarrow \mathrm{BN}$. Inserting $\mathrm{ReLU}(\cdot)$ after convolution is important for convergence. Our simulation implementation is based on Pytorch [93].

TABLE 2: Comparison with the state-of-the-art binary models using ResNet-18, ResNet-34, ResNet-50 and MobileNetV1 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)	BENN (6 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	61.0	65.1	67.0	**67.5**
	Top-5	89.4	67.1	73.2	79.5	85.9	-	85.8	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**
MobileNetV1	Top-1	70.6	-	-	58.2	-	-	66.5	68.7	-

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 [24]	2	2	64.9	85.9
DOREFA-Net [22]	2	2	62.6	84.4
SYQ [96]	1	8	62.9	84.6

5.1.2 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 [14], XNOR-Net [15], Bi-Real Net [82] and ABC-Net [45] and BENN [94]. 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 Section 4, 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 skip connections to high-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 [95]. 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. Moreover, 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.

5.1.3 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 [22], SYQ [96] and LQ-Nets [24]. As described in Section 4, 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 state-of-the-art 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.

5.1.4 Extension on MobileNetV1

We make two modifications to adapt our Group-Net to MobileNetV1. 1): The 3×3 depth-wise and the 1×1 point-wise convolutional blocks in the MobileNetV1 [13] are replaced by the 3×3 and 1×1 vanilla convolutions in parallel with shortcuts respectively. 2): For the reduction block, the input channels of the downsampling layers are twice the output channels, which violates the requirement for adding identity skip connections. Unlike previous works [82], [97] which adopts a 1×1 convolutional layer to match the dimension, we propose to duplicate the input activation and concatenate two blocks with the same inputs to address the channel number difference. We observe that Group-Net** with 5 bases only has 1.9% Top-1 accuracy drop.

5.2 Ablation study on ImageNet classification

5.2.1 Layer-wise vs. group-wise binary decomposition

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	**65.1**	**85.8**
LBD	5	57.6	79.7

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 [43], [44], [45], [46], [47] and the differences are described in Section 4. 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.

5.2.2 Effect of the soft gate

In this section, we further analysis the effect of soft gate described in Section 3.2. We show the convergence curves in Figure 6 and quantitative results in Table 5. From the results, we can observe consistent accuracy improvement for various architectures. This shows that increasing the gradient paths and learning the information flow within

BNNs is important for maintaining the performance. For instance, on ResNet-34, learning the soft gates can improve the Top-1 accuracy by 2.2%.

TABLE 5: The effect of soft gates on ImageNet.

Model		Full	Group-Net (w/o softgates)	Group-Net
ResNet-18	Top-1 %	69.7	64.2	**65.1**
	Top-5 %	89.4	85.0	**85.8**
ResNet-34	Top-1 %	73.2	66.3	**68.5**
	Top-5 %	91.4	86.2	**88.0**
ResNet-50	Top-1 %	76.0	67.5	**69.5**
	Top-5 %	92.9	86.7	**88.2**

5.2.3 Effect of the number of bases

TABLE 6: Validation accuracy (%) of Group-Net on ImageNet with different number of bases. All cases are based on the ResNet-18 network with binary weights and activations.

Model	Bases	Top-1	Top-5	Top-1 gap	Top-5 gap
Full-precision	1	69.7	89.4	-	-
Group-Net	1	56.4	79.5	13.3	9.9
Group-Net	3	62.5	84.2	7.2	5.2
Group-Net	5	**65.1**	**85.8**	**4.6**	**3.6**

We further explore the influence of number of bases K to the final performance in Table 6. When the number is set to 1, it corresponds to directly binarize the original full-precision network and we observe apparent accuracy drop compared to its full-precision counterpart. With more bases employed, we can find the performance steadily increases. The reason can be attributed to the better fitting of the floating-point structure, which is a trade-off between accuracy and complexity. It can be expected that with enough bases, the network should has the capacity to approximate the full-precision network precisely. With the multi-branch group-wise design, we can achieve high accuracy while still significantly reducing the inference time and power consumption. Interestingly, each base can be implemented using small resource and the parallel structure is quite friendly to FPGA/ASIC.

5.2.4 Effect of the group space

To show the importance of the structure design, we define more methods for comparison as follows: **GBD v1:** We implement with the group-wise binary decomposition strategy, where each base consists of one block. It corresponds to the approach described in Eq. (5) and is illustrated in Figure 7 (a). **GBD v2:** Similar to GBD v1, the only difference is that each group base has two blocks. It is illustrated in Figure 7 (b) and is explained in Eq. (6). **GBD v3:** It is an extreme case where each base is a whole network, which can be treated as an ensemble of a set of binary networks. This case is shown in Figure 7 (d).

We present the results in Table 7. We observe that learning the soft connections between each group results in the best performance on ResNet-18. And methods based on hard connections perform relatively worse. Moreover, different architectures can have a great impact on the performance. For example, the Top-1 gap between "GBD v2" and "GBD v3" is 3% even though their complexity is nearly the same. From the results, we can conclude that designing compact binary structure is essential for highly accurate classification.

TABLE 7: Comparisons between several group-wise decomposition strategies. Top-1 and Top-5 accuracy (%) gaps to the corresponding full-precision ResNet-18 network are also reported.

Model	Bases	Top-1	Top-5	Top-1 gap	Top-5 gap
Full-precision	1	69.7	89.4	-	-
Group-Net	5	**65.1**	**85.8**	**4.6**	**3.6**
GBD v1	5	63.0	84.8	6.7	4.6
GBD v2	5	62.2	84.1	7.5	5.3
GBD v3	5	59.2	82.3	10.5	7.1

5.2.5 Width multiplier vs. structure approximation

We further compare the "structure approximation" with the width multiplier baseline [13], which simply multiplies the channel number by a fixed ratio. To make the two settings directly comparable, we set $K = 4$ and the width multiplier to 2. The results are shown in Table 8.

TABLE 8: Performance of Group-Net and the width multiplier on ImageNet.

Model	Top-1 %	Top-5 %
Group-Net**	**66.3**	**86.6**
Group-Net	64.2	85.6
Width multiplier	63.5	85.2

We observe that Group-Net outperforms the width multiplier baseline on ResNet-18. This further justifies the structured approximation can better preserves the information.

5.3 Evaluation on semantic segmentation

In this section, we further evaluate Group-Net on the PASCAL VOC 2012 semantic segmentation benchmark [98] 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 [99], resulting in 10,582 training images. The performance is measured in terms of averaged pixel intersection-over-union (mIOU) over 21 classes.

Implementation details. Our experiments are based on FCN [3], where we adjust the dilation rates of the last two stages in ResNet with atrous convolution to make the output stride equal to 8. We empirically set dilation rates to be (4, 8) in last two stages. Similar to the structure of FCN-32s and FCN-16s, we define our modified baselines as FCN-8s-C5 and FCN-8s-C4C5, where C4 and C5 denote extracting feature from the final convolutional layer of the 4-th and 5-th stage, respectively. 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=0 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.

5.3.1 Experiments on FCN

The main results are reported in Table 9 and Table 10. From the results in Table 9, we can observe that when all bases using the same dilation rate, there is an obvious 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

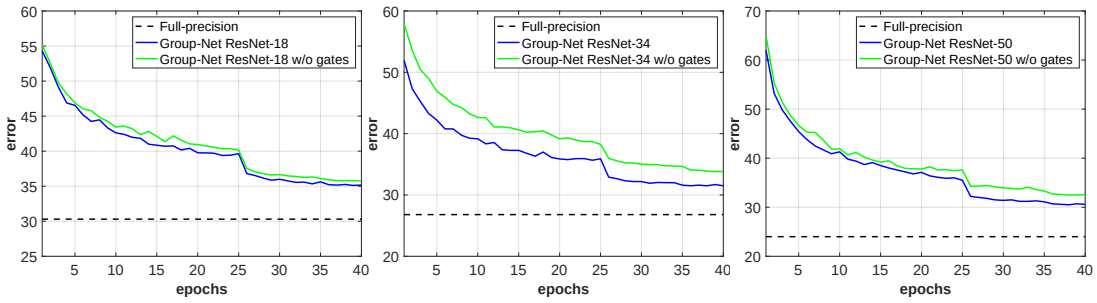


Fig. 6: Validation curves for ImageNet classification.

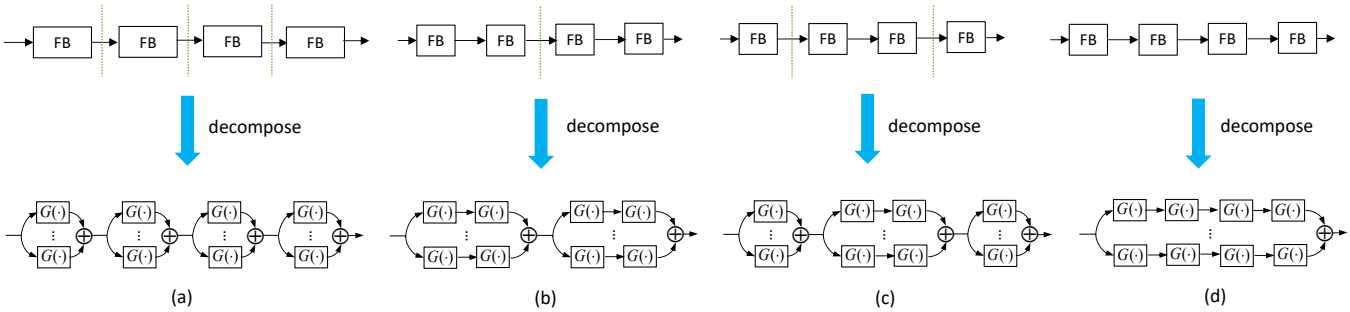


Fig. 7: Illustration of several possible group-wise architectures. We assume the original full-precision network comprises four blocks. "FB" represents the floating-point block. $G(\cdot)$ is defined in Section 3.2.2, which represents a binary block. We omit the skip connections for convenience. (a): Each group comprises one block and we approximate each floating-point block with a set of binarized blocks. (b): Decompose the network into groups, where each group contains two blocks. Then we approximate each floating-point group using a set of binarized groups. (c): Each group contains different number of blocks. (d): An extreme case. We directly decompose the whole floating-point network into an ensemble of several binary networks.

TABLE 9: Performance of Group-Net on PASCAL VOC 2012 validation set with FCN.

Backbone	Model	mIOU	Δ
ResNet-18, FCN-8s-C5	Full-precision	64.9	-
	LQ-Net (3-bit)	62.5	2.4
	Group-Net	60.5	4.4
ResNet-18, FCN-8s-C4C5	Full-precision	67.3	-
	LQ-Net (3-bit)	65.1	2.2
	Group-Net	62.7	4.6
ResNet-34, FCN-8s-C5	Full-precision	72.7	-
	LQ-Net (3-bit)	70.4	2.3
	Group-Net	68.2	4.5
ResNet-50, FCN-8s-C5	Full-precision	73.1	-
	LQ-Net (3-bit)	70.7	2.4
	Group-Net	68.3	4.8

TABLE 10: Performance of Group-Net with BPAC on PASCAL VOC 2012 validation set with FCN.

Backbone	Model	mIOU
ResNet-18, FCN-8s-C5	Full-precision (multi-dilations)	67.6
	Group-Net + BPAC	63.8
	Group-Net** + BPAC	**65.1**
ResNet-18, FCN-8s-C4C5	Full-precision (multi-dilations)	70.1
	Group-Net + BPAC	66.3
	Group-Net** + BPAC	**67.7**
ResNet-34, FCN-8s-C5	Full-precision (multi-dilations)	75.0
	Group-Net + BPAC	71.2
	Group-Net** + BPAC	**72.8**
ResNet-50, FCN-8s-C5	Full-precision (multi-dilations)	75.5
	Group-Net + BPAC	71.8
	Group-Net** + BPAC	**72.8**

segmentation performance. 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.

To reduce performance loss, we further employ diverse dilated rates on parallel binary bases to capture the multi-scale information without increasing any computational complexity. This formulates our final approach Group-Net + BPAC in Table 10, which shows significant improvement over the Group-Net counterpart. Moreover, the performance of Group-Net + BPAC is comparable with the full-precision baseline in Table 9, which strongly justifies the flexibility of Group-Net.

In addition, we can observe that Group-Net based on ResNet-50 backbone has the largest relative performance drop. It further shows the widely used bottleneck structure is not suited to BNNs as explained in Section 5.1.2.

5.3.2 *Experiments on DeepLabv3*

For training the DeepLabv3 baseline, we use Adam as the optimizer. The initial learning rate for training backbone network is set to 1e-4, and is multiplied by 10 for ASPP module. Similar to image classification, we keep the first layer and last classification layer to 8-bit. We employ the layer-wise approximation in quantizing the ASPP module. We set $K = 5$ for both backbone and ASPP. The training details are the same with those in FCN except that we use the polynomial decay of learning rate. The results are provided in Table 12.

Moreover, a problem still exists in training binary DeepLabv3. The ASPP module uses large dilation rates for the three 3×3 convolutions with $rates = \{12, 24, 36\}$ when *output_stride=8*. For training BNNs with Eq. (2), we apply

one-paddings to constrain activations to $\{-1, 1\}$. However, for atrous convolution with large rates, padding ones can introduce high bias and make the optimization difficult. To solve this problem, we instead binarize the activations to $\{0, 1\}$ following the quantizer in [22]. Note that the numerical difference is only a scalar whether activations are represented by $\{-1, 1\}$ or $\{0, 1\}$ in bitwise operations to accelerate the dot products [22], [24]. The importance for binarizing activations to $\{0, 1\}$ is shown in Table 11.

TABLE 11: The difference for binarizing ASPP with $\{-1, 1\}$ and $\{0, 1\}$. The metric is mIOU.

Model	full-precision	$\{0, 1\}$	$\{-1, 1\}$
ResNet-18	72.1	63.4	50.8

It is worth noting that the comparable counterpart of DeepLabv3 is the *Group-Net + BPAC* with FCN-8s-C5 in Table 10. The main difference is that in the proposed BPAC module, we directly incorporate the multiple dilation rates in the backbone network to capture multi-scale context. In contrast, DeepLabv3 embeds the ASPP module on top of the backbone network. With the same *output_stride*, the computational complexity of FCN is lower than DeepLabv3 since no additional ASPP is needed. With the simpler quantized FCN framework with BPAC, we can achieve comparable or even better performance than quantized DeepLabv3. For example, with the ResNet-34 backbone, GroupNet + BPAC outperforms DeepLabv3 by 4.4 w.r.t mIOU. It also shows that binarization on ASPP is sensitive to the final performance, since the binarization process constrains the feature magnitude to $\{-1, 1\}$ which causes the multi-scale information loss.

5.4 Evaluation on object detection

In this section, we evaluate Group-Net on the general object detection task. Our experiments are conducted on the large-scale detection benchmark COCO [100]. Following [76], [84], we use the COCO *trainval35k* split (115K images) for training and *minival* split (5K images) for validation. We also report our results on the *test_dev* split (20K images) by uploading our detection results to the evaluation server. In all settings, we set $K = 4$.

Implementation details. In specific, the backbone is initialized by the pretrained weights on ImageNet classification. Group-Net is then fine-tuned with Adam with the initial learning rate of 5e-4 and the batch size of 16 for 90,000 iterations. The learning rate is decayed by 10 at iteration 60,000 and 80,000, respectively. Note that we keep updating the BN layers rather than fix them during training. Other hyper-parameters are kept the same with [76], [79].

TABLE 12: Performance on the PASCAL VOC 2012 validation set with DeepLabv3.

Backbone	Model	mIOU	Δ
ResNet-18	Full-precision	72.1	-
	Backbone	**66.9**	**5.2**
	Backbone + ASPP	63.4	8.7
ResNet-34	Full-precision	74.4	-
	Backbone	**70.0**	**4.4**
	Backbone + ASPP	66.2	8.2
ResNet-50	Full-precision	76.9	-
	Backbone	**71.8**	**5.1**
	Backbone + ASPP	68.1	8.8

5.4.1 Performance evaluation

We report the performance on the COCO validation set in Table 15 and test set in Table 16, respectively. We can observe that Group-Net achieves promising results over all ResNet architectures. For instance, with the ResNet-18 backbone, the gap of AP is only 6.4 while we save considerable computational complexity. This strongly shows that the proposed Group-Net is a general approach that can be extended on many fundamental computer vision tasks. Furthermore, we highlight that we are the first to explore binary neural networks on object detection in the literature.

We compare with the state-of-the-art quantized object detector FQN [26], and also quantize the detector with LQ-Net using the proposed training strategy. We observe that GroupNet** outperforms the two comparing methods. It shows that learning independent heads is more effective than freezing batch normalization layers in FQN to stablize the optimization. Moreover, the better low-precision feature quality of GroupNet also contributes to superiority of the performance.

We further analysis the memory and computation saving in Table 13. Since we work on the quantization scheme, the floating-point complexity metric FLOPs is not suitable for quantization. Therefore, we also follow the metric in Uniq [101], where BOPs is used to measure the number of bit-operations in a neural network. For the 18-layer, 34-layer and 50-layer networks, Group-Net reduces the memory usage by $6.04\times$, $6.47\times$ and $6.40\times$, respectively, and it achieves BOPs reduction of $17.60\times$, $18.10\times$ and $18.77\times$ in comparison with the full-precision counterpart. Moreover, we directly count the total number of BOPs in the network. In practice, the binary bases are implemented in parallel and actual speed varies according to different hardware platforms.

5.4.2 Detection components analysis

We further analysis the affect of quantizing the backbone, feature pyramid and heads to the final performance, respectively. The results are reported in Table 14.

From Table 14, binarizing the backbone and the feature pyramid only downgrades the performance by a small margin. However, binarizing heads causes an obvious AP drop. It can be attributed to that heads are responsible for adapting the extracted multi-level features to the classification and regression objectives. As a result, its representability is crucial for robust detectors. However, the multi-level information is undermined when being constrained to $\{-1, 1\}$. *This shows that the detection modules other than the backbone are sensitive to quantization, and we leave it as our future work.*

By comparing the results with respect to weight sharing, we observe the original sharing heads strategy that widely used in full-precision detection frameworks performs extremely bad in the binary setting. In specific, with ResNet-18 backbone, the AP gap between with and without weight sharing reaches by 8.1. It is worth noting that separating the parameters does not increase any additional computational complexity. Even though the number of parameters are increased (i.e., by $\sim$ 4 times in heads), the memory consumption is still significantly reduced due to the 1-bit storage.

TABLE 13: Memory usage, FLOPs and BOPs on object detection.

Backbone	Model	Memory usage	Memory saving	FLOPs	FLOPs saving	BOPs	BOPs saving
ResNet-18	Full-precision	8.69×10^8 bit	1.0×	7.43×10^{10}	1.0×	8.36×10^{13}	1.0×
	Group-Net	1.44×10^8 bit	6.04×	5.65×10^9	13.15×	4.75×10^{12}	17.60×
ResNet-34	Full-precision	1.19×10^9 bit	1.0×	9.79×10^{10}	1.0×	1.10×10^{14}	1.0×
	Group-Net	1.84×10^8 bit	6.47×	7.12×10^9	13.75×	6.05×10^{12}	18.10×
ResNet-50	Full-precision	1.13×10^9 bit	1.0×	8.35×10^{10}	1.0×	1.16×10^{14}	1.0×
	Group-Net	1.76×10^8 bit	6.40×	6.22×10^9	13.42×	6.16×10^{12}	18.77×

TABLE 14: Performance on COCO validation set with different binarized components.

Backbone	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision	33.8	51.8	36.1	19.3	36.4	44.4
	Backbone	33.2	50.9	36.0	18.8	36.1	44.0
	Backbone + Feature pyramid	32.5	49.6	35.3	18.6	34.5	42.1
	Backbone + Feature pyramid + Heads (shared)	18.6	35.5	17.9	10.1	22.4	27.2
	Backbone + Feature pyramid + Heads (w/o shared)	28.9	45.3	31.2	15.4	30.5	38.1

TABLE 15: Performance on the COCO validation set based on FCOS.

Backbone	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision	33.8	51.8	36.1	19.3	36.4	44.4
	FQN	26.2	43.5	26.7	13.3	29.5	35.7
	LQ-Net	28.0	45.0	30.6	15.0	29.8	36.6
	Group-Net**	**28.9**	**45.3**	**31.2**	**15.4**	**30.5**	**38.1**
ResNet-34	Full-precision	37.5	55.9	40.3	22.6	40.8	47.4
	FQN	28.8	46.3	30.0	14.8	31.0	38.8
	LQ-Net	30.8	47.0	33.4	16.2	32.5	40.0
	Group-Net**	**31.5**	**47.6**	**33.8**	**16.9**	**32.3**	**40.1**
ResNet-50	Full-precision	38.6	57.4	41.4	22.3	42.5	49.8
	FQN	29.7	47.1	30.8	15.3	31.8	39.3
	LQ-Net	32.1	48.3	34.8	17.6	33.2	40.6
	Group-Net**	**32.7**	**49.0**	**35.5**	**17.8**	**33.6**	**41.4**

TABLE 16: Performance on the COCO test set based on FCOS.

Backbone	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision	33.9	51.9	36.4	19.4	35.6	42.2
	FQN	26.3	43.8	26.8	13.5	29.4	35.4
	LQ-Net	28.2	45.3	30.7	15.1	29.6	36.1
	Group-Net**	**29.1**	**45.7**	**31.4**	**15.5**	**30.2**	**37.5**
ResNet-34	Full-precision	37.8	56.3	41.0	22.1	40.1	46.7
	FQN	28.9	46.6	30.1	14.5	31.5	37.8
	LQ-Net	30.9	47.5	33.5	16.0	32.3	39.5
	Group-Net**	**31.6**	**47.8**	**33.7**	**16.6**	**32.8**	**39.7**
ResNet-50	Full-precision	38.8	57.9	41.9	22.4	41.5	48.0
	FQN	29.9	47.4	31.0	14.9	31.7	39.0
	LQ-Net	32.0	48.5	35.1	17.4	32.7	40.0
	Group-Net**	**32.8**	**49.4**	**35.6**	**18.2**	**33.4**	**40.6**

6 Acceleration on Hardware

To investigate the real execution speed of the proposed Group-Net and other related counterparts including fixed-point quantization and width-multiplier, we develop the acceleration code (GPU version only) on resource constrained platforms and provide the benchmark results. Experimental platforms include HiSilicon Kirin 970, Qualcomm 821 as well as Qualcomm 835. For fair comparison, we fix the frequency of the sub-systems (such as the CPU, GPU and DDR) if possible in order to prevent interference from the DVFS (dynamic voltage and frequency scaling) on the modern operating system. We fuse the batch normalization layers into the corresponding convolutional layers. Multiple rounds of experiments are conducted and the profiling data is averaged for statistic stability. We set the batch size to 1 in all scenarios.

We first compare the layer-wise execution time of Group-Net (refer to Eq. (3)) with 4 bases and 2-bit models. The results are listed in Table 17. A total of 11 cases are included in this experiment. All cases are with convolutions of 3×3 kernel, padding 1 and stride 1. For convenience, we configure the input and output feature maps to have the same shape, where input channels/width/height = output channels/width/height, respectively. For the first five cases, we fix the channel number to be 64 and increase the resolution from 28 to 448 with a multiplier 2. For the last six cases, we fix the resolution to be 56×56 and double the channel number from 16 to 512. From the results, we observe that the inference speed of Group-Net is slighter slower than that of the 2-bit model due to the extra addition operations discussed in Section 4.

We also report the overall speed for all quantized layers in Table 18 of BNNs, Group-Net (4 bases), 2-bit models. From Table 18, we first observe that the real execution time of Group-Net** and 2-bit models is less than $4\times$ slower than the binary case. Moreover, we also report the speedup of the 4 bases Group-Net against the 2-bit model. Specifically, the relative speedup ranges from 0.94 to 0.97 across different architectures (ResNet-18 and ResNet-34) on various devices (Qualcomm 821 and 835). It implies the extra addition operations analyzed in Section 4 has small impact (3% to 6%) on the overall inference speed. In summary, the structure approximation is still hardware-friendly but more flexible and accurate.

7 Conclusion

In this paper, we have explored 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 have also proposed to learn the decomposition automatically. Experimental results have proved the effectiveness of the proposed approach on the ImageNet classification task. More importantly, we have generalized the proposed Group-Net approach from image classification tasks to more challenging fundamental computer vision tasks, namely dense prediction tasks such as semantic segmentation and object detection. We highlight that we may be among the first few approaches to apply binary neural networks on general semantic segmentation and object detection tasks, and achieve encouraging performance on PASCAL VOC and COCO datasets with binary networks. Last but not least, we have developed the underlying acceleration code and speedup evaluation comparing with other quantization strategies is analyzed on several

TABLE 17: The execution time (us) of a single layer with different configurations.

Device	model	case1	case2	case3	case4	case5	case6	case7	case8	case9	case10	case11
Qualcomm 821	2-bit	**1928**	**3543**	**7968**	**24914**	**93069**	**1512**	**2348**	**3546**	**7188**	**21187**	**75486**
	Group-Net	2014	3621	8163	25376	94550	1613	2487	3627	7311	21362	75982
	Relative	4.4%	2.2%	2.4%	1.8%	1.6%	6.6%	5.9%	2.3%	1.7%	0.8%	0.7%
Qualcomm 835	2-bit	**2076**	**2969**	**6463**	**20528**	**73967**	**1600**	**1900**	**2852**	**5646**	**15822**	**69803**
	Group-Net	2255	3206	6930	21776	75010	1754	1998	2982	5746	17676	71140
	Relative	8.6%	8.0%	7.2%	6.1%	1.4%	9.6%	5.1%	4.6%	1.8%	11.7%	1.9%
Kirin 970	2-bit	**900**	**2234**	**2544**	8040	**31027**	**1048**	**1594**	**2415**	**2390**	**6750**	**24087**
	Group-Net	1017	2603	2550	**8030**	31145	1060	**1594**	2535	2478	6764	24326
	Relative	13.0%	16.5%	0.2%	-0.1%	0.4%	1.1%	0.0%	5.0%	3.7%	0.2%	1.0%

TABLE 18: Exact execution time (ms) and speedup ratios for overall quantized layers. We run 5 times and report the results with mean and standard deviation.

Device	Network	Binary	Group-Net**	2-bit	Binary vs. Group-Net**	Binary vs. 2-bit	Group-Net** vs. 2-bit
Q835	ResNet-18	12.1±0.2	47.6±0.3	44.8±0.2	3.93	3.70	0.94
	ResNet-34	25.3±0.3	95.7±0.5	90.6±0.4	3.78	3.58	0.95
Q821	ResNet-18	15.7±0.3	54.2±1.6	52.5±1.0	3.45	3.34	0.97
	ResNet-34	32.3±0.6	108.0±2.6	105.2±2.1	3.34	3.26	0.97

platforms, which serves as a strong benchmark for further research.

ACKNOWLEDGMENTS

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] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

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

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 3431–3440.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 801–818.

[5] 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.*, 2016, pp. 779–788.

[6] 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.*, 2015, pp. 91–99.

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

[8] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2, 2017, p. 6.

[9] 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, pp. 2790–2799.

[10] 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, pp. 7920–7928.

[11] 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.

[12] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 1251–1258.

[13] 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.

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

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 525–542.

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

[17] 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*. IEEE, 2004, p. 149.

[18] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. Int. Conf. Learn. Repren.*, 2018.

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

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

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

[22] 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.

[23] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5918–5926.

[24] 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, pp. 365–382.

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

[26] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, and R. Fan, "Fully quantized network for object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 2810–2819.

[27] Y. Wei, X. Pan, H. Qin, W. Ouyang, and J. Yan, "Quantization mimic: Towards very tiny cnn for object detection," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 267–283.

[28] 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.

[29] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018, pp. 6848–6856.

[30] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Repren.*, 2017.

[31] 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, pp. 4092–4101.

[32] 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, pp. 8697–8710.

[33] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural network for accurate image classification and semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 413–422.

[34] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

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

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

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

[38] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 4350–4359.

[39] E. Park, J. Ahn, and S. Yoo, "Weighted-entropy-based quantization for deep neural networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5456–5464.

[40] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 11 408–11 417.

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

[42] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 8612–8620.

[43] Y. Guo, A. Yao, H. Zhao, and Y. Chen, "Network sketching: Exploiting binary structure in deep cnns," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5955–5963.

[44] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao, "Performance guaranteed network acceleration via high-order residual quantization," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2017, pp. 2584–2592.

[45] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 344–352.

[46] J. Fromm, S. Patel, and M. Philipose, "Heterogeneous bitwidth binarization in convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4006–4015.

[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.*, 2017, pp. 2625–2631.

[48] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "Ai benchmark: Running deep neural networks on android smartphones," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 0–0.

[49] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "TVM: An automated end-to-end optimizing compiler for deep learning," in *USENIX Symp. Operating Systems Design & Implementation*, 2018, pp. 578–594.

[50] Y. Umuroglu, N. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*. ACM, 2017, pp. 65–74.

[51] H. Yang, M. Fritzsche, C. Bartz, and C. Meinel, "Bmxnet: An open-source binary neural network implementation based on mxnet," in *Proc. of the ACM Int. Conf. on Multimedia*. ACM, 2017, pp. 1209–1212.

[52] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. T. Shen, "TBN: Convolutional neural network with ternary inputs and binary weights," in *Proc. Eur. Conf. Comp. Vis.*, 2018.

[53] J. Zhang, Y. Pan, T. Yao, H. Zhao, and T. Mei, "dabnn: A super fast inference framework for binary neural networks on arm devices," in *Proc. ACM Int. Conf. Multimedia*, 2019, pp. 2272–2275.

[54] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Networks*, vol. 100, pp. 49–58, 2018.

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

[56] 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.*, 2016, pp. 2818–2826.

[57] 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.*, vol. 4, 2017, p. 12.

[58] 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.*, 2018, pp. 4510–4520.

[59] 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, pp. 19–34.

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

[61] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 269–284.

[62] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 552–568.

[63] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 1925–1934.

[64] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[65] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016, pp. 3194–3203.

[66] S. Chandra and I. Kokkinos, "Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 402–418.

[67] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 82–92.

[68] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 9126–9135.

[69] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 325–341.

[70] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network," *arXiv preprint arXiv:1811.11431*, 2018.

[71] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 1440–1448.

[72] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014, pp. 580–587.

[73] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 7263–7271.

[74] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[75] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 21–37.

[76] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2017, pp. 2980–2988.

[77] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 742–751.

[78] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 2820–2828.

[79] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2019.

[80] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 5187–5196.

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

[82] 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, pp. 722–737.

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

[84] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 2117–2125.

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

[86] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017, pp. 5987–5995.

[87] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Repren.*, 2019.

[88] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[89] 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.

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

[91] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comp. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

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

[93] 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.

[94] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019, pp. 4923–4932.

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

[96] 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, pp. 4300–4309.

[97] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," in *Proc. Int. Conf. Learn. Repren.*, 2020.

[98] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comp. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[99] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011, pp. 991–998.

[100] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comp. Vis.*, 2014, pp. 740–755.

[101] C. Baskin, E. Schwartz, E. Zheltonozhskii, N. Liss, R. Giryes, A. M. Bronstein, and A. Mendelson, "Uniq: Uniform noise injection for non-uniform quantization of neural networks," *arXiv preprint arXiv:1804.10969*, 2018.