NASB
Neural Architecture Search for Binary Convolutional Neural Networks

Zhu, Baozhou; Al-Ars, Zaid; Hofstee, H. Peter

DOI
10.1109/IJCNN48605.2020.9207674

Publication date
2020

Document Version
Accepted author manuscript

Published in
2020 International Joint Conference on Neural Networks, IJCNN 2020 - Proceedings

Citation (APA)
Zhu, B., Al-Ars, Z., & Hofstee, H. P. (2020). NASB: Neural Architecture Search for Binary Convolutional Neural Networks. In *2020 International Joint Conference on Neural Networks, IJCNN 2020 - Proceedings* (pp. 1-8). Article 9207674 (Proceedings of the International Joint Conference on Neural Networks). IEEE. https://doi.org/10.1109/IJCNN48605.2020.9207674

NASB: Neural Architecture Search for Binary Convolutional Neural Networks

Baozhou Zhu[1] Zaid Al-Ars[1] Peter Hofstee[1,2]

[1]*Delft University of Technology*
Delft, The Netherlands
{b.zhu-1, z.al-ars}@tudelft.nl

[2]*IBM Research*
Austin, TX, USA
hofstee@us.ibm.com

***Abstract*—Binary Convolutional Neural Networks (CNNs) have significantly reduced the number of arithmetic operations and the size of memory storage needed for CNNs, which makes their deployment on mobile and embedded systems more feasible. However, the CNN architecture after binarizing requires to be redesigned and refined significantly due to two reasons: 1. the large accumulation error of binarization in the forward propagation, and 2. the severe gradient mismatch problem of binarization in the backward propagation. Even though substantial effort has been invested in designing architectures for single and multiple binary CNNs, it is still difficult to find an optimal architecture for binary CNNs. In this paper, we propose a strategy, named NASB, which adopts Neural Architecture Search (NAS) to find an optimal architecture for the binarization of CNNs. Due to the flexibility of this automated strategy, the obtained architecture is not only suitable for binarization but also has low overhead, achieving a better trade-off between the accuracy and computational complexity of hand-optimized binary CNNs. The implementation of NASB strategy is evaluated on the ImageNet dataset and demonstrated as a better solution compared to existing quantized CNNs. With insignificant overhead increase, NASB outperforms existing single and multiple binary CNNs by up to 4.0% and 1.0% Top-1 accuracy respectively, bringing them closer to the precision of their full precision counterpart. The code and pretrained models will be publicly available.**

***Index Terms*—binary neural networks, neural architecture search, quantized neural networks, efficiency**

I. Introduction

With the increasing depth and width of Convolutional Neural Networks (CNNs), they have demonstrated many breakthroughs in a wide range of applications, such as image classification, object detection, and semantic segmentation [1]–[3]. However, the computation and storage overhead of deep and wide CNNs require millions of FLOPs and parameters, which hinders the real-time deployment on resource-constrained mobile and embedded platforms.

Numerous researchers proposed different approaches to address the efficiency problem of deploying CNNs, including low bit-width quantization [4], [5], network pruning [6], and efficient architecture design [2], [7]. Binarization [8], [9] is the most efficient quantization method among all those methods with reduced bit-widths, where a real-valued weight or activation is represented with a single bit and the multiplication and addition of a convolution can be implemented simply by XNOR and popcount bitwise operations, which is roughly 64 times faster to compute and requires 32 times less storage than their full precision counterpart. However, the extreme quantization method of single binary CNNs introduces the largest accumulation error in the forward propagation. In addition, during the backward propagation, its gradient flow is the most difficult to determine due to the high gradient mismatch problem [10] among all quantization methods with reduced bit-widths.

Existing published work focuses on improving the quantization quality mainly using value approximation and structure approximation. These two approximations are complementary to each other and could be exploited together. Value approximation seeks to find an optimized algorithm to quantize weights and activations while preserving the original network architecture. Knowledge distillation [11], [12] and loss-aware [13] objectives are introduced to find optimal local minima for quantized weights and activations. Advanced quantization functions [4], [5], [10] are proposed to minimize the quantization error between quantized values and their full precision counterparts. Tight approximation of the derivative of the non-differentiable activation function [9], [14] is explored to alleviate the gradient mismatch problem.

Unlike the above value approximation methods, structure approximation seeks to redesign the architecture of quantized CNNs to match the representational capacity of their original full precision counterpart. Structure approximation is more important for binary CNNs than for other low bit-width CNNs because binarization introduces the largest accumulation error and the severest gradient mismatch problem among all quantization methods with reduced bit-widths. Bi-Real Net [9] and Group-Net [15] are the state-of-the-art structure approximation methods for single and multiple binary CNNs, respectively. However, designing architectures for quantized CNNs is highly non-trivial especially for binary CNNs. In this paper, NASB strategy is proposed to automatically seek an optimal structure approximation for binary CNNs. In particular, this strategy uses Neural Architecture Search (NAS) to figure out an optimized architecture for the binarization of CNNs. After searching in a large space, the finalized CNN architecture is suitable for binarization, whose accuracy outperforms previous binary CNNs with insignificant computational complexity increase.

The main contributions in this paper are summarized as follows.

- We proposed the NASB strategy, which adopts NAS to automatically find an optimal architecture for the binarization of CNNs. Using NAS, NASB can search in a large space to figure out an optimized CNN architecture, which is suitable for binarization.
- Compared to the recent literature of binary CNNs, NASB achieves a sizable accuracy increase with negligible additional overhead, providing a better solution to address the trade-off between accuracy and efficiency.
- Our proposed NASB strategy is evaluated for ResNet on the ImageNet classification dataset, providing extensive experimental results to show the effectiveness of our proposal.

II. Related work

In this section, recent network quantization methods and efficient architecture design developments of CNNs are described.

A. Network quantization

There is substantial interest in research and development of dedicated hardware for CNNs to be deployed on embedded systems and mobile devices, which motivates the study of network quantization. Low bit-width approaches [4], [5], [16], [17] quantized weights and activations using fixed-point numbers, which reduces model size and compute time, but still requires multipliers to compute. Binary CNNs [8], [18], [19] are trained with weights and activations constrained to binary values $+1$ or -1, which can be categorized as single binary CNNs. The Ternary Weight Networks (TWN) [20] approach is proposed to reduce the loss of single binary CNNs by introducing 0 as the third quantized value, while Trained Ternary Quantization (TTQ) [21] enables the asymmetry and training of its scaling coefficients. However, the accuracy degradation of single binary and ternary CNNs is unacceptable for advanced CNNs like ResNet and large scale datasets like ImageNet. Multiple binary CNNs [15], [22]–[24] are promising attempts to reduce the accuracy gap between binary CNNs and their full precision counterpart. However, all the architectures of current single or multiple binary CNNs are human-designed. Further architecture optimization is possible using automated methods, such as [19], which encodes the number of channels in each layer, but does not change the operations and their connections in the model; something that we do consider in our proposed NASB strategy.

B. Efficient architecture design

Recently, more and more literature focuses on the efficient architecture design for the deployment of CNNs. Replacing 3×3 convolutional weights with 1×1 weights (in SqueezeNet [3] and GoogLeNet [25]) is suggested to decrease the computational complexity. Moreover, separable convolutions are adopted in Inception series [26] and further generalized as depthwise separable convolutions in Xception [27], MobileNet [2] and ShuffleNet [7]. Group convolution has been used as an efficient way to enhance efficiency in [7], [28], where the input activations and convolutional kernels are factorized into groups and executed independently inside each group. MobileNet [29] and ShuffleNet [30] series have been working on depthwise separable convolutions and shuffle operations to achieve a better trade-off between efficiency and accuracy. ESPNetv2 [31] uses group point-wise and depth-wise dilated separable convolutions to learn representations from a large effective receptive field, delivering state-of-the-art performance across different tasks. NAS [32]–[34] has demonstrated much success in automating network architecture design, achieving state-of-the-art efficiency [35], [36].

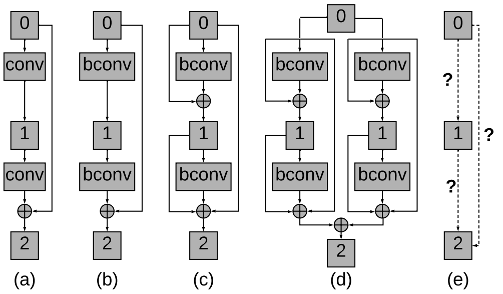


Fig. 1. Human-designed architecture for single and multiple binary CNNs. conv and bconv refer to full precision and binary convolutional layer, respectively, while Batch Normalization and the Relu layers are omitted.

III. Method

In this section, the problem of finding an architecture for the binarization of CNNs is defined and presented. Then, we explain NASB strategy, which can adopt the NAS technique to figure out an optimal architecture for binarizing CNNs. Last but not least, variants of NASB strategy are illustrated to enhance its efficiency.

A. Problem definition

Given a full precision convolutional cell, what is an optimal architecture for binarizing it? The accumulation error in the forward propagation of binarization is the largest and the gradient flow in the backward propagation is the most difficult to take care of among all quantization methods with different bit-widths. As a result, it is essential to figure out an optimized architecture for binarizing CNNs. Here, this convolutional cell can be a convolutional layer, block, group, and network.

There have been various attempts to answer the above question, as shown in Fig. 1. Fig. 1(a) is a full precision convolutional block. Fig. 1(b), (c), and (d) describe the proposed architecture in the literature representing XNOR [8], Bi-Real [9], and Group-Net [15], respectively, where the scaling coefficients have been omitted. Although lots of the above human efforts have been dedicated for designing an architecture for single and multiple binary CNNs, it is still worth to explore an optimal convolutional cell architecture using the automatical approach as represented by the question marks in Fig. 1(e).

The question can be expressed as a directed acyclic graph in Fig. 1(e), which represents an ordered sequence of 3 nodes and 3 edges with one operation for each edge. The number of nodes, edges, and operations for each edge can be freely selected. Each node x^i represents a feature map and each edge (i, j) is associated with several operations $o^{i,j}$ to transform x^i. Here the convolutional cell has one input and output node, and its output is obtained by addition of all intermediate nodes. In the following, the binarization and NAS techniques adopted in this paper are presented.

a) Binary convolutional neural networks: Given a full precision convolutional layer, its inputs, weights and outputs are denoted as $I \in R^{N\times C_{in}\times H\times W}$, $W \in R^{C_{in}\times C_{out}\times h\times w}$ and $O \in R^{N\times C_{out}\times H\times W}$, respectively, where N, C_{in}, C_{out}, H, W, h and w refer to the batch size, the number of input and output channels, the height and width of the feature maps, and the height and width of the weights, respectively.

Using the binarization method of weights in [8], we approximate the full precision weights W as binary weights b^W with the sign of W and the scaling coefficient s, where the scaling coefficient is computed as the mean of the absolute values of W. Adopting the Straight Through Estimator (STE) [37], the forward and backward propagations of the weights binarization are shown as follows.

$$\begin{aligned}\text{Forward: } & b^W = s \times \text{sign}(W)\\ \text{Backward: } & \frac{\partial L}{\partial W} = \frac{\partial L}{\partial b^W} \times \frac{\partial b^W}{\partial W} \approx s \times \frac{\partial L}{\partial b^W}\end{aligned} \tag{1}$$

where L is the total loss.

Using the binarization method of activations in [9], we approximate the full precision activations as binary activations b^I by a piecewise polynomial function. The forward and backward propagations of the activations binarization can be written as follows.

$$\begin{aligned}\text{Forward: } & b^I = \text{sign}(I)\\ \text{Backward: } & \frac{\partial L}{\partial I} = \frac{\partial L}{\partial b^I} \times \frac{\partial b^I}{\partial I}\\ \text{where} & \frac{\partial b^I}{\partial I} = \begin{cases} 2 + 2I, -1 \leq I < 0 \\ 2 - 2I, 0 \leq I < 1 \\ 0, \text{otherwise} \end{cases}\end{aligned} \tag{2}$$

b) Gradient based neural architecture search: We adopt a gradient-based NAS in [38]. To reduce memory footprint during training the over-parameterized network, we use the strategy from [34] to binarize and learn the M real-valued architecture parameters α_i.

In the forward propagation, the M real-valued architecture parameters α_i are transformed to the real-valued path weights p_i, and then to the binary gates g_i as follows.

$$p_i = \frac{\exp(\alpha_i)}{\sum\limits_{j=1}^{M} \exp(\alpha_j)} \tag{3}$$

$$g_i = \text{binarize}(p_i) = \begin{cases} 1, \text{with probability } p_i \\ 0, \text{with probability } (1 - p_i) \end{cases} \tag{4}$$

In the backward propagation, the STE [37] is also applied.

$$\frac{\partial L}{\partial p_j} \approx \frac{\partial L}{\partial g_j} \tag{5}$$

The gradient w.r.t. architecture parameters can be estimated as follows.

$$\begin{aligned}\frac{\partial L}{\partial \alpha_i} &= \sum_{j=1}^{M} \frac{\partial L}{\partial p_j}\frac{\partial p_j}{\partial \alpha_i} \approx \sum_{j=1}^{M} \frac{\partial L}{\partial g_j}\frac{\partial p_j}{\partial \alpha_i}\\ &= \sum_{j=1}^{M} \frac{\partial L}{\partial g_j} p_j(\delta_{ij} - p_i)\end{aligned} \tag{6}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

B. NASB strategy

In this section, we present the details about how NASB strategy works. To apply NAS for binarizing CNNs, the key innovation is to leverage the NAS technique to find a NASB-convolutional cell as an optimal architecture for binarizing their full precision counterpart, where the NASB-convolutional cell can be a replacement for a binarized convolutional layer, block, group, and network. NASB strategy consists of the following stages: searching stage, pretraining stage, and fine-tuning stage. In the following, the search space of a NASB-convolutional cell in NASB strategy is described, including its connections and operations. Besides, its training algorithm is presented.

a) Connections of a NASB-convolutional cell: Taking that we are exploring an optimal architecture for a convolutional group as an example, the connections of a NASB-convolutional cell in NASB strategy is explored at the searching stage as shown in Fig. 2.

Fig. 2(a) describes all the connections of a NASB-convolutional cell during the training of the searching stage, which consists of a backbone and a NAS-convolutional cell. The left cell is the backbone of the NASB-convolutional cell, which is a standard convolutional group in ResNet [1]. The right cell is considered as a NAS-convolutional cell, which is a directed acyclic graph consisting of 5 nodes, 10 edges, and 10 operations for every edge. Here 5 nodes are used to keep the layer depth of a NASB-convolutional cell in NASB strategy the same as its full precision counterpart, which will not increase the latency during inference. The connections of the backbone are fixed and there is no need to specify architecture parameters for it. During the training of the searching stage, the model weights of the NASB-convolutional cell and architecture parameters of the NAS-convolutional cell can be updated alternately, and only one operation on every edge in the NAS-convolutional cell is sampled and active at every step. In this way, the inactive paths reduce the memory requirements.

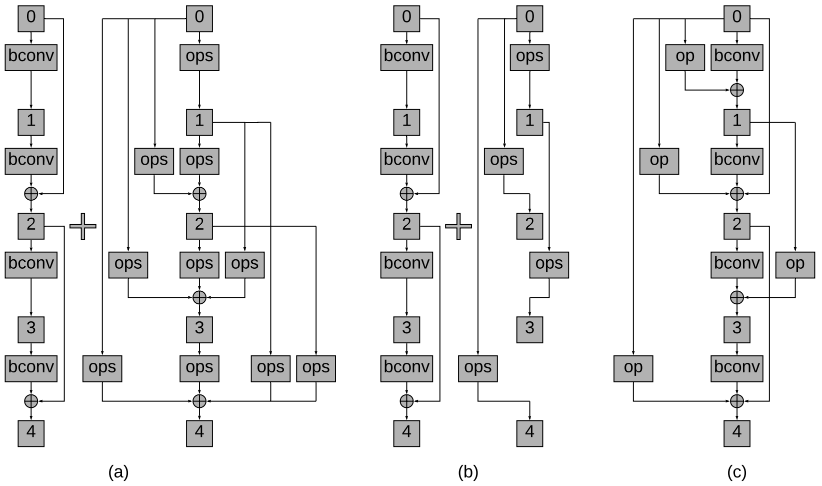


Fig. 2. Exploring connections of a NASB-convolutional cell at the searching stage. conv and bconv refer to full precision and binary convolutional layer, respectively. ops refers to a set of operations as shown in Fig. 3, among which one operation is active during the training of the searching stage.

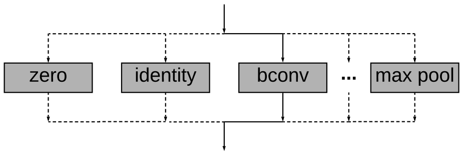


Fig. 3. A set of operations in every ops.

Fig. 2(b) is the finalized architecture after completing the training of the searching stage. In the NAS-convolutional cell, we retain only one predecessor for every node and one operation for every edge except for the node with the number 0. Fig. 2(c) is a more compact representation of Fig. 2(b), showing the output of every node in the NASB-convolutional cell (except for the node with the number 0) defined as the addition of the two inputs from the backbone and the NAS-convolutional cell.

b) Operations of a NASB-convolutional cell: Taking the number of bitwise operations and binary parameters of a 3×3 binary convolution as 1 unit, the number of bitwise operations and binary parameters of all the operations used in NASB strategy are unified as shown in Table I. The overhead of Batch Normalization and Relu layer is not included.

TABLE I
THE NUMBER OF BITWISE OPERATIONS AND BINARY PARAMETERS OF THE OPERATIONS USED IN NASB. F AND B REFER TO FULL PRECISION AND BINARY PRECISION, RESPECTIVELY. BO AND BP REFER TO BITWISE OPERATIONS AND BINARY PARAMETERS, RESPECTIVELY.

Operations	Bo	Bp
op0 = Zero (F)	0	0
op1 = 3×3 average pooling (F)	< 1	0
op2 = 3×3 max pooling (F)	< 1	0
op3 = Identity (F)	0	0
op4 = 1×1 convolution (B)	$1/9$	$1/9$
op5 = 3×3 convolution (B)	1	1
op6 = 5×5 convolution (B)	$25/9$	$25/9$
op7 = 1×1 dilated convolution (B)	$1/9$	$1/9$
op8 = 3×3 dilated convolution (B)	1	1
op9 = 5×5 dilated convolution (B)	$25/9$	$25/9$

The number of bitwise operations and binary parameters of the binary convolution is $NC_{out}HW \times 2C_{in}hw$ and $C_{out}C_{in}hw$, respectively, when no bias is added. Scaling the kernel size of the binary convolution by a scaling coefficient of s_k, both the number of bitwise operations and binary parameters are scaled by s_k^2. Changing the dilation rate will not increase the number of bitwise operations and binary parameters of the binary convolution, when the additional cost introduced by padding is omitted. The number of bitwise operations required for computing every individual output of the binary convolution is approximately $2C_{in}hw$, while the number of bitwise operations required for computing every individual output of a 3×3 max and average pooling is $8d$ and $16d$, respectively, where d is the bit-width of pooling operations and $2C_{in}hw \gg 16d$ in general. Besides, pooling will not introduce any parameters.

c) Three-stage training algorithm: As shown in Algorithm 1, the training algorithm of NASB strategy consists of three stages: the searching stage, pretraining stage, and finetuning stage. The goal of the searching stage is to get an optimal binary CNN architecture, which is done by using NAS to train a binary CNN model M_s from scratch on dataset D. The pretraining stage is used to train a full precision CNN model M_p from scratch on dataset D', whose architecture is finalized from the searching stage. The finetuning stage is used to binarize the pre-trained CNN obtained from the pretraining stage and finetune it on dataset D' to get a binary CNN model M_f.

The binary CNN model finalized from the searching stage is the same as model M_f used in the finetuning stage except for some minor differences because of their different datasets. Performing the searching stage on a small dataset D rather

Algorithm 1 Three-stage training algorithm

```
Input: Dataset $D = \{(X_i, Y_i)\}_{i=1}^{S}$ for the searching stage,
    dataset $D' = \{(X_i', Y_i')\}_{i=1}^{S}$ for the pretraining and fine-
    tuning stages.
Output: Binary CNN model $M_s$ for the searching stage, full
    precision CNN model $M_p$ for the pretraining stage, and
    binary CNN model $M_f$ for the finetuning stage.
  Stage 1: The searching stage
for $epoch = 1$ to $L$ do
    for $batch = 1$ to $T$ do
        Randomly sample a mini-batch validation data from
        $D$, freeze the model weights of model $M_s$, and
        update its architecture parameters.
        Randomly sample a mini-batch training data from $D$,
        freeze the architecture parameters of model $M_s$, and
        update its model weights.
    end for
end for
  Stage 2: The pretraining stage
for $epoch = 1$ to $L$ do
    for $batch = 1$ to $T$ do
        Randomly sample a mini-batch training data from $D'$
        and update the weights of model $M_p$.
    end for
end for
  Stage 3: The finetuning stage
for $epoch = 1$ to $L$ do
    for $batch = 1$ to $T$ do
        Randomly sample a mini-batch training data from $D'$
        and update the weights of model $M_f$.
    end for
end for
```

than directly on target dataset D' can be regarded as a proxy task to find the optimal binary architecture model M_f for the finetuning stage, which can enable a large search space and significantly accelerate the computation of NASB strategy. After binarizing the full precision CNN model M_p from the pretraining stage, we directly get the binary CNN model M_f for the finetuning stage.

C. Variants of NASB strategy

In this section, a number of variants of NASB strategy are presented to improve the accuracy over state-of-the-art multiple binary CNNs. Taking NASB ResNet18 as an example, there are 4 NASB-convolutional cells, and each of them is composed of 5 nodes. we retain only one predecessor for every node and one operation for every edge except for the node with the number 0. By changing the number of NASB-convolutional cells and operations for every node, different variants of NASB strategy are explored.

NASBV1 strategy enlarges the search space of a NASB-convolutional cell. In NASBV1 ResNet18, there are 2 NASB-convolutional cells, and each of them is composed of 9 nodes. In NASBV2 ResNet18, we adopt the method in [38] to retain 4 operations instead of 1 operation for the output node of the NASB-convolutional cell. In NASBV3 ResNet18, We copy all the NASB-convolutional cells once to get two binary branches. The two branches can be parallelized thoroughly except that we merge the information of the two branches at the end of every block using addition operation as in [15]. All the NASB-convolutional cells are different from each other, which can explore the optimal binary architecture for every NASB-convolutional cell. In NASBV4 ResNet18, we retain 4 operations (except for identity) instead of 1 operation for every node of the NASB-convolutional cell. In NASBV5 ResNet18, we retain 8 operations for the output node and 6 operations for the other nodes of the NASB-convolutional cell. Fig. 2(a) is the connections of a NASB-convolutional cell at the searching stages for the NASB strategy and the NASBV5 strategy. Fig. 4 is the derived architecture of the NASBV5 strategy after the searching stage.

TABLE II
ACCURACY OF NASB RESNET18 VARIANTS

Variants	Top-1	Top-5
NASB ResNet18	60.5%	82.2%
NASBV1 ResNet18	60.3%	82.3%
NASBV2 ResNet18	61.1%	82.7%
NASBV3 ResNet18	62.8%	84.1%
NASBV4 ResNet18	65.3%	85.9%
NASBV5 ResNet18	66.6%	87.0%

TABLE III
COMPARISONS OF RESNET18 WITH MULTIPLE BINARY METHODS.

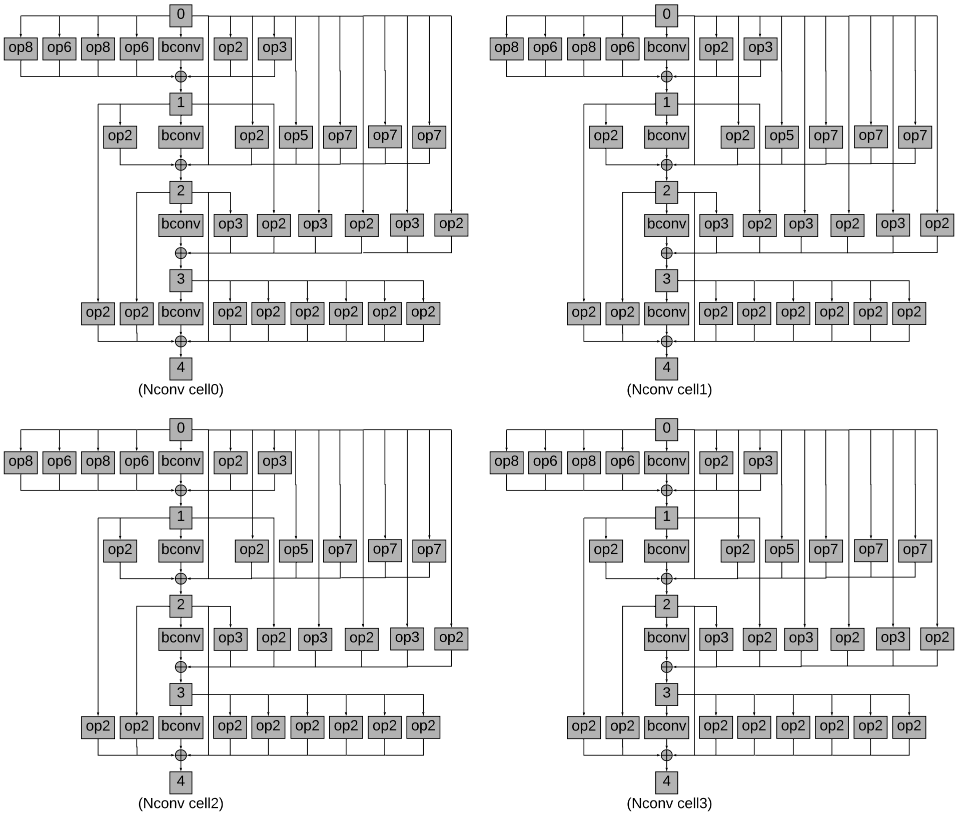
Model	Top-1	Top-5
Full precision	69.7%	89.4%
ABC-Net ($M = 5$, $N = 5$)	65.0%	85.9%
Group-Net (4 bases)	64.2%	85.6%
Group-Net** (4 bases)	66.3%	86.6%
NASBV4	65.3%	85.9%
NASBV5	66.6%	87.0%

IV. EXPERIMENTAL RESULTS ON IMAGENET DATASET

We applied our proposed NASB strategy for the binarization of ResNet [1], trained and evaluated on the ILSVRC2012 classification dataset [39]. ResNet is one of the most popular and advanced CNNs.

A. Implementation details

During the searching stage, we train model M_s on CIFAR-10. Half of the CIFAR-10 training data is used as a validation set. The Relu layer is not added in the searching stage. We train model M_s for 100 epochs with batch size 64. We use momentum SGD and Adam to optimize the model weights and architecture parameters, respectively. The experiments are performed on one GPU. In NASB ResNet18 and NASB ResNet34, all NASB-convolutional cells adopt 4 nodes and they use 3 nodes for NASB ResNet50. Due to memory limitations, we remove convolutions and dilated convolutions with kernel size 3 and 5 for NASB ResNet50 during this stage.


0
op8 op6 op8 op6 bconv op2 op3
1
op2 bconv op2 op5 op7 op7 op7
2
bconv op3 op2 op3 op2 op3 op2
3
op2 op2 bconv op2 op2 op2 op2 op2 op2
4
0
op8 op6 op8 op6 bconv op2 op3
1
op2 bconv op2 op5 op7 op7 op7
2
bconv op3 op2 op3 op2 op3 op2
3
op2 op2 bconv op2 op2 op2 op2 op2 op2
4
0
op8 op6 op8 op6 bconv op2 op3
1
op2 bconv op2 op5 op7 op7 op7
2
bconv op3 op2 op3 op2 op3 op2
3
op2 op2 bconv op2 op2 op2 op2 op2 op2
4
0
op8 op6 op8 op6 bconv op2 op3
1
op2 bconv op2 op5 op7 op7 op7
2
bconv op3 op2 op3 op2 op3 op2
3
op2 op2 bconv op2 op2 op2 op2 op2 op2
4

(Nconv cell0)

(Nconv cell1)

(Nconv cell2)

(Nconv cell3)

Fig. 4. Architecture of NASB-convolutional cells in NASBV5 ResNet18. Nconv cell refers to NASB-convolutional cell. conv and bconv refer to full precision and binary convolutional layer, respectively.

TABLE IV
COMPARISONS WITH SINGLE BINARY CNNS

Model		Full	BNN	XNOR	Bi-Real	NASB
ResNet18	Top-1	69.7%	42.2%	51.2%	56.4%	60.5%
	Top-5	89.4%	67.1%	73.2%	79.5%	82.2%
ResNet34	Top-1	73.2%	–	–	62.2%	64.0%
	Top-5	91.4%	–	–	83.9%	84.7%
ResNet50	Top-1	76.0%	–	–	62.6%	65.7%
	Top-5	92.9%	–	–	83.9%	85.8%

TABLE V
COMPARISONS OF RESNET18 WITH FIXED-POINT QUANTIZATION METHODS.

Model	W	A	Top-1	Top-5
Full precision	32	32	69.7%	89.4%
Dorefa-Net	2	2	62.6%	84.4%
SYQ	1	8	62.9%	84.6%
Lq-Net	2	2	64.9%	85.9%
NASBV4	1	1	65.3%	85.9%

During the pretraining stage, we train model M_p obtained from the last searching stage on the ILSVRC2012 classification dataset. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted. We do not apply any more sophisticated data augmentation to the training data. We use standard single-crop testing for evaluation. We insert the Relu layer and use the layer order as Conv→Relu→BN, and the $tanh$ function is applied to activation after the Batch Normalization layer.

During the finetuning stage, we binarize and train the pretrained model M_p from the pretraining stage into model M_f. The weights and activations are binarized using the method described in Section III-A0a. We keep 1×1 convolution to full-precision in this stage. We adopt Adam as the optimizer and set weight decay to 0 since the binarization can be recognized as a kind of regularization.

B. Experimental results of NASB variants

The accuracy of different variants is compared in Table II. The accuracy of NASBV1 ResNet18 is almost the same as that of NASB ResNet18. We conjecture that $28/36$ of the total edges in NASBV1 ResNet18 is removed rather than $6/10$ of the total edges in NASB ResNet18, which will change model M_s too much and remedy the benefits of a larger search space. For other variants of NASB strategy, we observe the increased operations of NASB-convolutional cell bring Top-1 accuracy improvement by up to 6.0%. It is expected that with more operations retained, NASB variants can achieve higher accuracy. We present the finalized architecture of four NASB-convolutional cells in NASBV5 ResNet18, as shown in Fig. 4,

TABLE VI
MEMORY USAGE AND FLOPS CALCULATION OF BI-REAL NET, GROUP-NET, NASB NET, AND FULL PRECISION MODELS

Model	Memory usage	Memory saving	Flops	Speedup
Bi-Real ResNet18	33.6Mbit	11.14 ×	1.63×10^8	11.06 ×
NASB ResNet18	33.8Mbit	11.07 ×	1.71×10^8	10.60 ×
ResNet18	374.1Mbit	–	1.81×10^9	–
Bi-Real ResNet34	43.7Mbit	15.97 ×	1.93×10^8	18.99 ×
NASB ResNet34	44.0Mbit	15.86 ×	2.01×10^8	18.26 ×
ResNet34	697.3Mbit	–	3.66×10^9	–
Bi-Real ResNet50	176.8Mbit	4.62 ×	5.45×10^8	7.08 ×
NASB ResNet50	178.1Mbit	4.60 ×	6.18×10^8	6.26 ×
ResNet50	817.8Mbit	–	3.86×10^9	–
ABC-Net ($M = 5$, $N = 5$) ResNet18	72.3Mbit	5.17 ×	6.74×10^8	2.70 ×
Group-Net (4 bases) ResNet18	62.1Mbit	6.03 ×	2.62×10^8	6.90 ×
Group-Net** (4 bases) ResNet18	83.9Mbit	4.46 ×	3.38×10^8	5.35 ×
NASBV4 ResNet18	70.7Mbit	5.30 ×	2.81×10^8	6.45 ×
NASBV5 ResNet18	88.3Mbit	4.24 ×	3.52×10^8	5.15 ×
ResNet18	374.1Mbit	–	1.81×10^9	–

which is derived from Fig. 2(a) after the searching stage. In this figure, we retain 8 operations for the output node and 6 operations for the other nodes of every NASB-convolutional cell.

C. Comparisons with the state-of-the-art quantized CNNs

As shown in Table IV, Table III, and Table V, we compare our NASB strategy with single binary CNNs, multiple parallel binary CNNs, and fixed-point CNNs using different quantization methods, respectively. All the comparison results are directly cited from the corresponding papers.

As shown in Table IV, Bi-Real Net [9] is the state-of-the-art single binary CNNs. Compared with Bi-Real ResNet with varying layers from 18 to 50, our proposed NASB ResNet show consistent accuracy improvement by 4.1%, 1.8%, and 3.1% Top-1 accuracy respectively.

As shown in Table III, we compare our NASB strategy with ABC-Net and Group-Net, which is a multiple binary CNN and can be implemented in a parallel way. Both NASBV4 and NASBV5 achieve higher accuray than ABC-Net. NASBV4 and NASBV5 show better accuracy performance than Group-Net and Group-Net** by 1.1% and 0.3%, respectively.

As shown in Table V, Lq-Net is the current best-performing fixed-point method. Multiple binary CNNs with K binary branches are preferable to fixed-point CNNs with $\sqrt{K}$ bit-width considering the computational complexity and memory bandwidth [15]. Thus, NASBV4 with 4 operations retained for every node requires less overhead while still achieves better accuracy.

D. Computational complexity analysis

To analyze the computational complexity of our proposed NASB strategy, we compare with Bi-Real Net, Group-Net, and full precision models in terms of memory usage saving and computation speedup as shown in Table VI.

The memory usage is computed as the summation of the number of real-valued parameters times 32 bit and the number of binary parameters times 1 bit. We use Flops to measure the computation and assume that bitwise XNOR and popcount operations can be calculated in parallel of 64 on current CPUs. Thus, the Flops is calculated as the summations of $1/64$ of the number of bitwise operations and the number of real-valued operations. Following the suggestion from [8], [9], [15], we keep the first convolutional layer, the last fully connected layer, and the downsampling layer as full precision.

Bi-Real Net [9] can be seen as a suboptimal binary CNN architecture of our NASB Net, where one edge connected to its last node is retained for every node and one identity operation remains for every edge. The finalized NAS-convolutional cells in NASB ResNet18 includes 12 max pooling and 4 identity operations, and they are composed of 20 max pooling and 12 identity operations in NASB ResNet34. In NASB Res50, the NAS-convolutional cells consist of 41 max pooling, 6 identity, and 1 1x1 dilated convolution operations. Compared to Bi-Real Net, the increased computational complexity is mainly due to max pooling. The Flops or the number of bitwise operations of a 3x3 max pooling is less than that of a 3x3 convolution, and the additional number of trainable parameters introduced by Batch Normalization of max pooling operation is $2C_{out}$.

As shown in Table VI, both the additional memory usage and Flops of NASB ResNet of varying depths are negligible compared to Bi-Real Net. ABC-Net requires much more Flops than Group-Net and NASB variants. The increased memory usage and Flops of NASB V5 and NASB V4 ResNet18 are insignificant compared to Group-Net** and Group-Net respectively.

V. CONCLUSION

In this paper, we proposed a NASB strategy to find an accurate architecture for binary CNNs. Specifically, the NASB strategy uses the NAS technique to identify an optimal architecture in a large search space, which is suitable for binarizing CNNs. We use the ImageNet classification dataset to prove the effectiveness of our proposed approach. With insignificant overhead increases, NASB strategy and its variants achieve up to 4.0% and 1.0% Top-1 accuracy improvement compared with the state-of-the-art single and multiple binary CNNs, respectively, providing a better trade-off between accuracy and efficiency. It is worth to worth to clarify that without we can easily extend our proposed NASB strategy to fixed-point quantized convolutional neural networks and other models for computer vision tasks beyond image classification, which can be explored further in the future.

Acknowledgment

This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[3] 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.

[4] 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.

[5] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.

[6] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.

[7] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[9] 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.

[10] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5918–5926.

[11] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[12] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.

[13] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *arXiv preprint arXiv:1611.01600*, 2016.

[14] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "Bnn+: Improved binary network training," *arXiv preprint arXiv:1812.11800*, 2018.

[15] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural networks for accurate image classification and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309.

[17] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350–4359.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[19] M. Shen, K. Han, C. Xu, and Y. Wang, "Searching for accurate binary neural architectures," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[20] F. Li and B. Liu, "Ternary weight networks," *CoRR*, vol. abs/1605.04711, 2016.

[21] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[22] J. Fromm, S. Patel, and M. Philipose, "Heterogeneous bitwidth binarization in convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 4006–4015.

[23] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems*, 2017, pp. 345–353.

[24] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4923–4932.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[26] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[28] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2752–2761.

[29] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," *arXiv preprint arXiv:1905.02244*, 2019.

[30] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[31] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9190–9200.

[32] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[33] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[34] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HylVB3AqYm

[35] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[36] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

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

[38] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[39] 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," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.