Towards lossless binary convolutional neural networks using piecewise approximation

Zhu, Baozhou; Al-Ars, Zaid; Pan, Wei

DOI
10.3233/FAIA200286

Publication date
2020

Document Version
Final published version

Published in
ECAI 2020: 24TH EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE

ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200286

Towards Lossless Binary Convolutional Neural Networks Using Piecewise Approximation

Baozhou Zhu [1] and **Zaid Al-Ars** [2] and **Wei Pan** [3]

Abstract. Binary Convolutional Neural Networks (CNNs) can significantly reduce the number of arithmetic operations and the size of memory storage, which makes the deployment of CNNs on mobile or embedded systems more promising. However, the accuracy degradation of single and multiple binary CNNs is unacceptable for modern architectures and large scale datasets like ImageNet. In this paper, we proposed a Piecewise Approximation (PA) scheme for multiple binary CNNs which lessens accuracy loss by approximating full precision weights and activations efficiently, and maintains parallelism of bitwise operations to guarantee efficiency. Unlike previous approaches, the proposed PA scheme segments piece-wisely the full precision weights and activations, and approximates each piece with a scaling coefficient. Our implementation on ResNet with different depths on ImageNet can reduce both Top-1 and Top-5 classification accuracy gap compared with full precision to approximately 1.0%. Benefited from the binarization of the downsampling layer, our proposed PA-ResNet50 requires less memory usage and two times Flops than single binary CNNs with 4 weights and 5 activations bases. The PA scheme can also generalize to other architectures like DenseNet and MobileNet with similar approximation power as ResNet which is promising for other tasks using binary convolutions. The code and pretrained models will be publicly available.

1 Introduction

CNNs have emerged as one of the most influential neural network architectures to tackle large scale machine learning problems in image recognition, natural language processing, and audio analysis [9, 13]. At the same time, their deployment on mobile devices and embedded systems are gaining more and more attention due to the increasing interest from industry and academia [12, 28]. However, the limited storage and computation resources provided by these platforms are an obstacle that is being addressed by numerous researchers working to reduce the complexity of CNNs [10, 35, 17, 30]. Fixed-point CNNs [24, 1, 36, 34, 37, 6] achieve even no accuracy loss with a suitable selection of bit-width, but the multiplication and the overflow processing of addition require considerable overhead. Binary CNNs have been demonstrated as a promising technique to make the deployment of CNNs feasible [3, 32, 4, 21]. In single binary CNNs, full precision weights and activations are binarized into 1 bit, so the multiplication and addition of the convolution are transformed into simple bitwise operations, resulting in significant storage and computation requirements reduction [25]. The accuracy degradation of the recently enhanced single binary CNN [23] is still high (12.9% Top-1 and 9.7% Top-5 accuracy degradation for ResNet18 on ImageNet) since much information has been discarded during binarization. ABC-Net [20] is the first multiple binary CNN, which shows encouraging result (around 5% Top-1 and Top-5 accuracy degradation for ResNet on ImageNet). [7, 8, 31, 19] calculate a series of binary values and their corresponding scaling coefficients through minimizing the residual error recursively, but they can not be paralleled. [39] propose Group-Net to explore structure approximation, and it is a complimentary approximation to value approximation. Multiple binary CNNs can be considered as a moderate way of quantization, that is much more accurate than single binary CNNs and more efficient than fix-point CNNs. But, there is still a considerable gap between full precision implementations and multiple binary CNNs, despite the fact that an unlimited number of weights and activation bases can be used.

To further reduce the gap between the full precision and multiple binary CNNs, we proposed Piece-wise Approximation (PA) scheme in this paper. Our main contributions are summarized as follows.

- PA scheme segments the whole range of the full precision weights and activations into many pieces and uses a scaling coefficient to approximate each of them, which can maintain parallelism of bitwise operation and lessen accuracy loss.
- With less overhead, our scheme achieves much higher accuracy than ABC-Net, which indicates that it provides a better approximation for multiple binary CNNs. Benefited from the binarization of the downsampling layer, our proposed PAResNet50 requires less memory usage and two times Flops than Bi-Real Net with 4 weights and 5 activations bases, which shows its potential efficiency advantage over single binary CNNs with a deeper network.
- With the increase of the number of the weight and activation bases, our proposed PA scheme achieves the highest classification accuracy for ResNet on ImageNet among all state-of-the-art single and multiple binary CNNs.

2 Related work

In this Section, we describe the forward propagation and backpropagation of typical schemes to quantize CNNs. In addition, the advantages and disadvantages of these quantized CNNs are discussed concerning efficiency and accuracy.

2.1 Single Binary Convolutional Neural Networks

In single binary convolutional neural networks [3, 32, 4, 29, 21], weights and activations are constrained to a single value $+1$ or -1.

[1] Delft University of Technology,The Netherlands,email:b.zhu-1@tudelft.nl
[2] Delft University of Technology,The Netherlands,email:z.al-ars@tudelft.nl
[3] Delft University of Technology,The Netherlands,email:wei.pan@tudelft.nl

The deterministic binarization function is described as follows.

$$x^b = \begin{cases} +1, x^r \geq 0 \\ -1, x^r < 0 \end{cases} \tag{1}$$

where x^b is the binarized variable, and x^r is the real-valued variable. During the backpropagation, the "Straight-Through Estimator" (STE) method [2] is adapted to calculate the derivatives of the binarization functions as follows, where C is the loss function.

$$\frac{\partial C}{\partial x^r} = \frac{\partial C}{\partial x^b} \tag{2}$$

Single binary CNNs is the most efficient quantization scheme among all the quantization schemes described in this paper. But, its accuracy degradation is too high to be deployed in practice.

2.2 Ternary Convolutional Neural Networks

In ternary convolutional neural networks [18, 38, 33, 11], ternary weights are used to reduce the accuracy loss of single binary CNNs by introducing 0 as the third quantized value, as follows.

$$x^t = \begin{cases} x^p : x^r > \Delta \\ 0 : |x^r| \leq \Delta \\ -x^n : x^r < -\Delta \end{cases} \tag{3}$$

where x^p and x^n are the positive and negative scaling coefficients, respectively, and Δ is a threshold to determine the ternarized variable x^t. During the backpropagation, the STE method is still applied.

Although the introduction of 0 improves the accuracy of single binary CNNs, it is still unacceptable to be deployed especially while training advanced CNNs on large scale dataset.

2.3 Fixed-point Convolutional Neural Networks

In fixed-point convolutional neural network [37, 34, 6, 15], weights, activations, and gradients are quantized using fixed-point numbers of different bit-widths. Taking the weights as an example, the quantization works as follows.

$$x^f = 2 \cdot \texttt{quantize}_f\left(\frac{\tanh(x^r)}{2\max(|\tanh(x^r)|)} + \frac{1}{2}\right) - 1 \tag{4}$$

where $\texttt{quantize}_f$ function quantizes the real-valued number x^r to the f-bit fixed-point number x^f. During the backpropagation, the STE method still works.

With a configuration of different bit-widths for the weights, activations, and gradients, the accuracy degradation of DoReFa-Net can be preserved and controlled. But, fixed-point multipliers result in the most substantial overhead among that of all the quantization schemes in this paper.

2.4 Multiple Binary Convolutional Neural Networks

In multiple binary convolutional neural networks [20, 7, 8, 31, 19, 39], a combination of multiple binary bases is adopted to approximate full precision weights and activations. Following is the weights approximation using linear combination.

$$x^r = \sum_{i=1}^{P} \varepsilon_i D_i \tag{5}$$

where ε_i is a trainable scaling coefficient and D_i is a binary (-1 and $+1$) weight base. During the backpropagation, STE method is still used.

The adoption of multiple binary bases in ABC-Net can lessen accuracy loss compared to single binary CNNs and maintain efficiency by using parallel bitwise operations compared to fix-point CNNs. Unfortunately, there is still a considerable gap between ABC-Net and full precision although as many as needed weight and activation binary bases can be used.

3 Piecewise approximation scheme

In this section, the PA scheme for multiple binary CNNs is illustrated, including the approximations of weights and activations. Also, the training algorithm and the inference architecture of PA-Net are clarified.

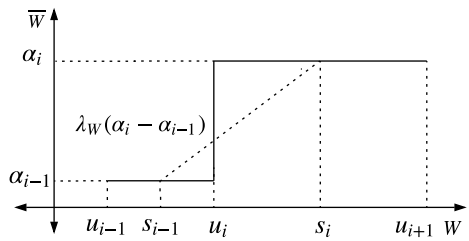


Figure 1: A sample of the forward propagation and backpropagation of weights approximation

3.1 Weights approximation

Since approximating weights channel-wise needs much more computational resources during training, we approximated weights as a whole in this paper.

The real-valued weights are $W \in R^{h \times w \times c_{in} \times c_{out}}$, where h, w, c_{in} and c_{out} represent the height and width of a filter, the number of input and output channels, respectively. In the forward propagation of PA scheme, these are estimated by $\overline{W}$, which is a piecewise function composed of the following $M+1$ pieces.

$$W \approx \overline{W} = \begin{cases} \alpha_1 \times B_W, W_j \in [-\infty, u_1] \\ \alpha_i \times B_W, W_j \in [u_{i-1}, u_i], i \in [2, \frac{M}{2}] \\ 0.0 \times B_W, W_j \in [u_{\frac{M}{2}}, u_{\frac{M}{2}+1}] \\ \alpha_i \times B_W, W_j \in [u_i, u_{i+1}], \\ \qquad\qquad i \in [\frac{M}{2}+1, M-1] \\ \alpha_M \times B_W, W_j \in [u_M, +\infty] \end{cases} \tag{6}$$

where u_i and α_i are the endpoint and scaling coefficient of the pieces, respectively. W_j is a scalar and a single weight of the tensor W. $W_j \in [-\infty, u_1]$ refers to the j^{th} weight of the tensor W which is in the range of $[-\infty, u_1]$. B_W is a tensor with all the values equal to 1.0 and has the same shape as W. Since the distribution of the weights is close to Gaussian, all the endpoints of the weights are fixed using $mean(W)$ and $std(W)$, which refer to the mean and standard deviation of the full precision weights, respectively. The M endpoints are almost uniformly sampled from $-2.0 \times std(W)$ to $2.0 \times std(W)$ except those near 0.0. To set the endpoints of the weights properly, we attempted some different settings, where the performance difference is negligible. Taking $M = 8$ as an example, we directly recommend the endpoints set as listed in Table 1.

Table 1: Endpoints of the weights with $M = 8$

Variables	u_1	u_2	u_3	u_4
Values ($\times$ $std(W)$)	-1.5	-1.0	-0.5	-0.25
Variables	u_5	u_6	u_7	u_8
Values ($\times$ $std(W)$)	0.25	0.5	1.0	1.5

Except for the $(\frac{M}{2}+1)$-*th* piece, the mean of all the full precision weights of every piece serves as the optimal estimation of its scaling coefficient.

$$\begin{cases} \alpha_1 = \texttt{reduce\_mean}(W), W_j \in [-\infty, u_1] \\ \alpha_i = \texttt{reduce\_mean}(W), W_j \in [u_{i-1}, u_i], i \in [2, \frac{M}{2}] \\ \alpha_i = \texttt{reduce\_mean}(W), W_j \in [u_i, u_{i+1}], \\ \qquad\qquad i \in [\frac{M}{2}+1, M-1] \\ \alpha_M = \texttt{reduce\_mean}(W), W_j \in [u_M, +\infty] \end{cases} \tag{7}$$

During the backpropagation, the relationship between $\overline{W}$ and W has to be established, and the whole range of the weights is segmented into M pieces.

$$\frac{\partial \overline{W}}{\partial W} = \begin{cases} \lambda_W(\alpha_2 - \alpha_1), W_j \in [-\infty, s_1] \\ \lambda_W(\alpha_{i+1} - \alpha_i), W_j \in [s_{i-1}, s_i], \\ \qquad\qquad i \in [2, \frac{M}{2}-1] \\ \lambda_W(0.0 - \alpha_{\frac{M}{2}}), W_j \in [s_{\frac{M}{2}-1}, s_{\frac{M}{2}}] \\ \lambda_W(\alpha_{\frac{M}{2}+1} - 0.0), W_j \in [s_{\frac{M}{2}}, s_{\frac{M}{2}+1}] \\ \lambda_W(\alpha_{i+1} - \alpha_i), W_j \in [s_i, s_{i+1}], \\ \qquad\qquad i \in [\frac{M}{2}+1, M-2] \\ \lambda_W(\alpha_M - \alpha_{M-1}), W_j \in [s_{M-1}, +\infty] \end{cases} \tag{8}$$

where s_i is the endpoint of the pieces. λ_W is a hyper-parameter, which is different when a different number of weight pieces is used. The endpoint s_i can be determined simply as follows

$$s_i = (u_{i+1} + u_i)/2.0, i \in [1, M-1] \tag{9}$$

The forward propagation while $W_j \in [u_{i-1}, u_{i+1}]$ and backpropagation while $W_j \in [s_{i-1}, s_i]$ are presented in Figure 1, where a linear function with slope $\lambda_W(\alpha_i - \alpha_{i-1})$ is used to approximate the piecewise function during the backpropagation.

3.2 Activations approximation

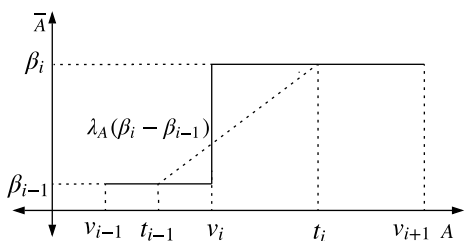


Figure 2: A sample of the forward propagation and backpropagation of activations approximation

To utilize bitwise operation for convolution, activations should be binarized as well. However, the distribution of the activations will vary in the inference stage which motivates us to apply batch normalization [14]. Batch normalization is applied before the approximation of the activations to force them to have zero mean and unit standard deviation.

The real-valued input activations are $A \in R^{n \times h \times w \times c_{in}}$, where n, h, w and c_{in} refer to batch size, height, width and number of channels, respectively. In the forward propagation of PA scheme, these are estimated by $\overline{A}$, which is a piecewise function composed of the following $N+1$ pieces.

$$A \approx \overline{A} = \begin{cases} 0.0 \times B_A, A_j \in [-\infty, v_1] \\ \beta_i \times B_A, A_j \in [v_i, v_{i+1}], i \in [1, N-1] \\ \beta_N \times B_A, A_j \in [v_N, +\infty] \end{cases} \tag{10}$$

where v_i and β_i are the endpoint and scaling coefficient of the pieces, respectively. $A_j \in [v_i, v_{i+1}]$ refers to the activations of matrix A which are in the closed range of $[v_i, v_{i+1}]$. B_A is a tensor with all the values equal to 1.0 and has the same shape as A. Both the endpoint v_i and the scaling coefficient β_i are trainable to learn the statistical features of the full precision activations. The bounded activation function is omitted since the endpoints are initialized with positive values.

During the backpropagation, the relationship between $\overline{A}$ and A has to be established, and the whole range of the activations is segmented into $N+2$ pieces.

$$\frac{\partial \overline{A}}{\partial A} = \begin{cases} 0.0, A_j \in [-\infty, t_0] \\ \lambda_A \times (\beta_1 - 0.0), A_j \in [t_0, t_1] \\ \lambda_A \times (\beta_{i+1} - \beta_i), A_j \in [t_i, t_{i+1}], \\ \qquad\qquad i = 1, ..., N-1 \\ 0.0, A_j \in [t_N, +\infty] \end{cases} \tag{11}$$

where t_i is the endpoint of the pieces. λ_A is a hyper-parameter, which is the same for all the layers in a given CNN and is different between different CNNs with different depths as used this paper. The endpoint t_i can be determined as follows

$$\begin{cases} t_i = (v_i + v_{i+1})/2.0, i = 1, ..., N-1 \\ t_0 = 2.0 \times v_0 - s_1 \\ t_N = v_N + \lambda_\Delta \end{cases} \tag{12}$$

where λ_Δ is a hyper-parameter, which is the same for all the layers in a given CNN and is different between different CNNs in this paper.

The forward propagation while $A_j \in [v_{i-1}, v_{i+1}]$ and backpropagation while $A_j \in [t_{i-1}, t_i]$ are presented in Figure 2, where a linear function with slope $\lambda_A(\beta_i - \beta_{i-1})$ is used to approximate the piecewise function during the backpropagation.

The scaling coefficient β_i is updated as follows

$$\frac{\partial C}{\partial \beta_i} = \frac{\partial C}{\partial \overline{A}} \frac{\partial \overline{A}}{\partial \beta_i} = \begin{cases} \texttt{reduce\_sum}(\frac{\partial C}{\partial \overline{A}}), A_j \in [v_i, v_{i+1}], i \in [1, N-1] \\ \texttt{reduce\_sum}(\frac{\partial C}{\partial \overline{A}}), A_j \in [v_N, +\infty], i = N \end{cases} \tag{13}$$

Similarly, the endpoint v_i is updated as

$$\frac{\partial C}{\partial v_i} = \frac{\partial C}{\partial \overline{A}} \frac{\partial \overline{A}}{\partial v_i} = \begin{cases} \lambda_A(\beta_1 - 0.0) \times \texttt{reduce\_sum}(\frac{\partial C}{\partial \overline{A}}), \\ \qquad\qquad A_j \in [t_0, t_1], i = 1 \\ \lambda_A(\beta_i - \beta_{i-1}) \times \texttt{reduce\_sum}(\frac{\partial C}{\partial \overline{A}}), \\ \qquad\qquad A_j \in [t_{i-1}, t_i], i \in [2, N] \end{cases} \tag{14}$$

Algorithm 1 Training a L-layer multiple binary CNN by PA scheme

Input: A mini-batch of inputs A_0 and targets A^*, weights W. Learning rate η, learning rate decay factor λ. The number of endpoints M, scaling coefficient α_i and endpoint u_i for weights, the number of endpoints N, scaling coefficient β_i and endpoint v_i for activations. PA is short for Piecewise Approximation scheme.

Output: Updated scaling coefficient β_i, endpoint v_i, weights W and learning rate η.

1. Computing the parameter gradients:

1.1. Forward path:

for $k = 1$ to L **do**
  $\overline{W} \leftarrow PA(W, u_i, \alpha_i, M)$
  $A \leftarrow Conv(\overline{A}, \overline{W})$
  if $k < L$ **then**
    $\overline{A} \leftarrow PA(A, v_i, \beta_i, N)$
  end if
end for

1.2. Backward propagation:

for $k = L$ to 1 **do**
  if $k < L$ **then**
    $(g_A, g_{v_i}, g_{\beta_i}) \leftarrow Back\_PA(g_{\overline{A}}, A, v_i, \beta_i, N)$
  end if
  $(g_{\overline{A}}, g_{\overline{W}}) \leftarrow Back\_Conv(g_A, \overline{A}, \overline{W})$
  $g_W \leftarrow Back\_PA(g_{\overline{W}}, W, u_i, \alpha_i, M)$
end for

2. Accumulating the parameter gradients:

for $k = 1$ to L **do**
  $\beta_i \leftarrow update(\beta_i, \eta, g_{\beta_i})$
  $v_i \leftarrow update(v_i, \eta, g_{v_i})$
  $W \leftarrow update(W, \eta, g_W)$
  $\eta \leftarrow \lambda\eta$
end for

3.3 Training algorithm

A sample of the training algorithm of PA-Net is presented as Algorithm 1, where details like batch normalization and pooling layers are omitted. SGD with momentum or ADAM [16] optimizer can be used to update parameters. Since our PA scheme approximates full precision weights and activations, using pre-trained models serves as initialization.

3.4 Inference architecture

Regarding the inference implementation of PA-Net, the latency is one of the most important metrics to be considered. Fortunately, the piecewise approximated weights or activations can be viewed as a linear combination of multiple binary bases ($+1$ and 0), which indicates a parallel inference architecture.

In the forward propagation, the approximated weights are represented as follows.

$$\overline{W} = \sum_{i=1}^{M} \alpha_i T_i \tag{15}$$

where T_i is a binary weight base, given as

$$T_i = \begin{cases} \begin{cases} B_W, W_j \in [-\infty, u_1] \\ 0.0 \times B_W, W_j \notin [-\infty, u_1] \end{cases} & , i = 1 \\ \begin{cases} B_W, W_j \in [u_{i-1}, u_i] \\ 0.0 \times B_W, W_j \notin [u_{i-1}, u_i] \end{cases} & , i \in [2, \frac{M}{2}] \\ \begin{cases} B_W, W_j \in [u_i, u_{i+1}] \\ 0.0 \times B_W, W_j \notin [u_i, u_{i+1}] \end{cases} & , \\ & i \in [\frac{M}{2} + 1, M - 1] \\ \begin{cases} B_W, W_j \in [u_M, +\infty] \\ 0.0 \times B_W, W_j \notin [u_M, +\infty] \end{cases} & , i = M \end{cases} \tag{16}$$

Similarly, the approximated activations in the forward propagation are expressed as follows.

$$\overline{A} = \sum_{i=1}^{N} \beta_i V_i \tag{17}$$

where V_i is a binary activation base, given as

$$V_i = \begin{cases} \begin{cases} B_A, A_j \in [v_i, v_{i+1}] \\ 0.0 \times B_A, A_j \notin [v_i, v_{i+1}] \end{cases} & , i = 1, ..., N-1 \\ \begin{cases} B_A, A_j \in [v_N, +\infty] \\ 0.0 \times B_A, A_j \notin [v_N, +\infty] \end{cases} & , i = N \end{cases} \tag{18}$$

Combined with the approximated weights, the forward propagation of the real-valued convolution can be approximated by computing $M \times N$ parallel bitwise convolutions. It is worth to notice that $\alpha_i\beta_j$ will be merged as one new scaling coefficient ϕ_k during the inference stage so that we omit their multiplication.

$$\begin{aligned} Conv(W, A) &\approx Conv(\overline{W}, \overline{A}) = Conv(\sum_{i=1}^{M} \alpha_i T_i, \sum_{j=1}^{N} \beta_j V_j) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N} \alpha_i\beta_j Conv(T_i, V_j) = \sum_{k=1}^{M\times N} \phi_k Conv(T_i, V_j) \end{aligned} \tag{19}$$

Taking $M = 3$ and $N = 3$ as an example, both the weights and activations use 3 bits to approximated their full precision counterpart. A full precision convolution can be computed with 9 parallel bitwise operations and 3 comparators, as shown in Figure 3, where the latency cost is as small as that of single binary CNNs. On the left is the structure of the activations approximation using binary activation bases V_1, V_2, and V_3. On the right is the structure of the weights approximation using binary weight bases T_1, T_2, and T_3. Thus, we implement the overall block structure of the convolution in the PA scheme with 9 parallel bitwise operations. It is worth to notice that computing the binary convolution blocks in this figure can be directly completed by AND and popcount operations, and the binary convolution blocks do not consist of Batch Normalization or Relu layer.

3.5 Efficiency analysis of different binary values

To the best of our knowledge, this is the first time to use binary values $+1$ and 0 instead of binary values -1 and $+1$ for single or multiple binary CNNs, and we present their efficiency analysis in terms of costumed hardware FPGA/ASIC.

When binary convolutions are computed by bitwise operation with binary values 0 and $+1$, the dot product of two bit-vectors x and y is computed using bitwise operations as follows.

$$x \cdot y = \text{bitcount}(\text{AND}(x, y)), x_i, y_i \in \{0, +1\} \forall_i \tag{20}$$

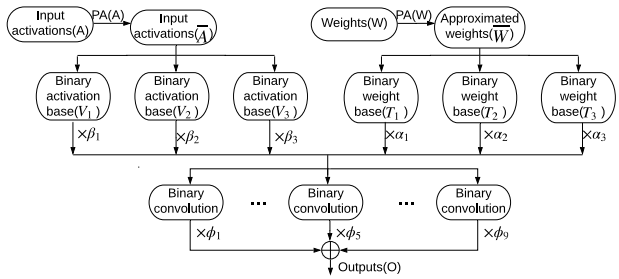


Figure 3: Parallel inference architecture of convolution in PA-Net

where $bitcount$ counts the number of bits in a bit-vector.

Similarly, when binary convolutions are computed by bitwise operation with binary values -1 and $+1$, the dot product of two bit-vectors x and y is computed using bitwise operations as follows.

$$x \cdot y = N - 2 \times \text{bitcount}(\text{XNOR}(x, y)), x_i, y_i \in \{-1, +1\} \forall_i \tag{21}$$

where N is the number of bits in a bit-vector.

Table 2: 2-input 7-nm CMOS gates propagation delay, area, and power

Items	Propagation delay $[ps]$	Active area $[nm^2]$	Power $[nW]$
XNOR	10.87	2.90×10^3	1.23×10^3
AND	9.62	1.45×10^3	6.24×10^2

In Table 2, we present the area footprint, the input to output propagation delay and the power consumption for 2-input Boolean gates using a commercial 7-nm FinFET technology (supply voltage $V_{DD} = 0.7V$). The active area and power consumption cost of an XNOR gate are two times as large as those of an AND gate, which indicates that the area and power consumption cost of a binary convolution with binary values -1 and $+1$ are two times as large as those of a binary convolution with binary values 0 and $+1$ (except for bitcount operation).

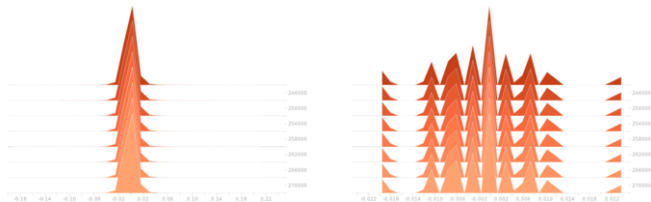
4 Experimental results on ImageNet dataset

We first trained and evaluated ResNet [9] using our proposed PA scheme on ImageNet ILSVRC2012 classification dataset [27]. Then we generalize our scheme to other CNN architectures such as DenseNet and MobileNet. Finally, the computational complexity of PA-Net is analyzed on CPUs and customized hardware.

We set the batch size of all our implementations to 64 due to the limit on available time and resources, which slightly limits the accuracy of the results. However, the accuracy is expected to increase with a larger batch size.

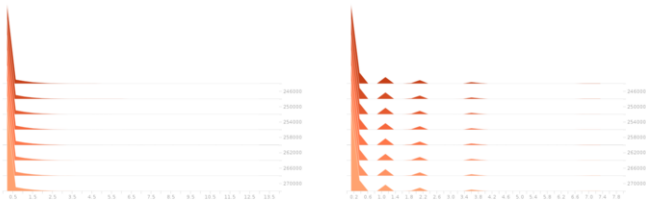
4.1 Weights and activations approximations

Using the ResNet18/group2/block1/conv1 layer, we sampled full precision weights and their approximations with $M = 8$. Their histograms are shown in Figure 4a and 4b, respectively. Horizontal axis and longitudinal axis represent the values and the number of values of weights/activations, respectively. Similarly, the comparison of activation histograms are shown in Figure 5, which are acquired from the ResNet18/group2/block1/conv2 layer and include the full precision activations in Figure 5a and their approximation with $N = 5$ in Figure 5b. As the comparisons show, the distributions of the approximated weights and activations are similar to those of the full precision weights and activations, respectively, which means that PA scheme provides an accurate way for multiple binary bases to approximate the distribution of their full precision counterparts.



(a) Full precision weights (b) Approximated weights

Figure 4: Distribution of full precision and approximated weights.



(a) Full precision activations (b) Approximated activations

Figure 5: Distribution of full precision and approximated activations.

4.2 Comparison with ABC-Net

Both PA-Net and ABC-Net can utilize parallel bitwise operation and achieve higher accuracy than single binary CNNs, so the differences between them need to be analyzed. The accuracy comparisons between PA-Net and ABC-Net are shown in Table 3.

Table 3 shows that PA-Net achieves higher accuracy than ABC-Net while requiring less overhead, which strongly supports the idea that PA-Net provides a better approximation than ABC-Net for both the weights and activations. In addition, Table 3 shows the unique advantage of PA-Net over ABC-Net since PA-Net can give higher accuracy for multiple binary CNNs by increasing M and N. However, we also re-implemented ABC-Net and reproduced the results, which shows that its accuracy remains unchanged (or even becomes worse) as we keep increasing M and N more than 5.

For the weights approximation only (i.e., when N is full precision), PA-ResNet18 gives no Top-1 accuracy loss with $M = 8$. PA-ResNet achieves higher accuracy with $M = 4$ and $N = 5$ than ABC-ResNet with $M = 5$ and $N = 5$, which means that PA-ResNet provides better approximation with less overhead. PA-ResNet with $M = 8$ and $N = 7$ reduce the Top-5 accuracy gap around 1.0%. But the accuracy of ABC-Net remains unchanged or even becomes worse with the increase of M and N more than 5 based on our re-implementation. PA-Net is expected to reach no accuracy loss with the increase of M and N, which we have not attempted due to the limitations of computational resources, training time, and the slow increase trend of accuracy with the increase of M and N.

4.3 Generalization to other CNN architectures

To demonstrate the generalization of PA scheme, we applied it on 1.0 MobileNet-224 [12] and DenseNet121 [13]. The results are

Table 3: Comparison with ABC-Net using ResNet as backbones

Model	M	N	$Top-1$	$Top-5$	$Top-1\ gap$	$Top-5\ gap$
ABC-ResNet18	5	full precision	68.3%	87.9%	1.0%	1.3%
PA-ResNet18	4	full precision	68.4%	88.3%	0.9%	0.9%
PA-ResNet18	8	full precision	69.3%	88.9%	0.0%	0.3%
ABC-ResNet18	5	5	65.0%	85.9%	4.3%	3.3%
PA-ResNet18	4	5	66.6%	87.1%	2.7%	2.1%
PA-ResNet18	8	7	68.1%	88.1%	1.2%	1.1%
ResNet18	full precision	full precision	69.3%	89.2%	–	–
ABC-ResNet34	5	5	68.4%	88.2%	4.9%	3.1%
PA-ResNet34	4	5	70.1%	89.2%	3.2%	2.1%
PA-ResNet34	8	7	71.5%	90.0%	1.8%	1.3%
ResNet34	full precision	full precision	73.3%	91.3%	–	–
ABC-ResNet50	5	5	70.1%	89.7%	6.0%	3.1%
PA-ResNet50	4	5	73.0%	91.0%	3.1%	1.8%
PA-ResNet50	8	7	74.3%	91.9%	1.8%	0.9%
ResNet50	full precision	full precision	76.1%	92.8%	–	–

Table 4: Generalization to DenseNet and MobileNet.

Model	M	N	$Top-1$	$Top-5$	$Top-1\ gap$	$Top-5\ gap$
PA-DenseNet121	8	6	72.3%	90.8%	2.7%	1.5%
DenseNet121	full precision	full precision	75.0%	92.3%	–	–
PA-1.0 MobileNet-224	8	7	69.0%	88.4%	1.6%	1.5%
1.0 MobileNet-224	full precision	full precision	70.6%	89.9%	–	–

shown in Table 4. Due to memory limitation, we implemented PA-DenseNet121 with $N = 6$. Its Top-1 accuracy loss is 2.7%, which is expected to decrease further with increasing N. Top-1 accuracy loss of PA-1.0 MobileNet-224 achieves 1.6% with $N = 7$. Pointwise convolution is binarized while depthwise convolution is kept as full precision convolution since they do not need significant computational resources.

4.4 Generalization to object detection

We choose SSD300 with the backbone network of ResNet50 as our baseline. The training dataset is VOC2007 + 2012, while the testing dataset is VOC2007 [5]. In the SSD300 model, we use the layers from Conv1 to Conv5_x of the pre-trained ResNet50 as the backbone network, apply residual blocks as the extra layers, and keep the number of feature maps the same as the original implementation [22]. All the backbone layers except Conv1 are binarized, while all the convolutional layers of the head network remain in full precision. We train the full precision ResNet50 on the ImageNet classification dataset as the backbone network, and then the full precision object detector SSD300 using the pre-trained ResNet50. Finally, we binarize and finetune the pre-trained object detector SSD300 with the PA scheme.

Table 5: Performances of the full-precision SSD300 network and its binary counterpart.

Detector	Backbone	Weights	Activations	mAP@0.5
SSD300	ResNet50	Full precision	Full precision	74.35
SSD300	ResNet50	$M = 4$	Full precision	72.53
SSD300	ResNet50	$M = 4$	$N = 5$	58.60

Applying the PA scheme to the SSD300 network, we present the results in Table 5. When only weights are binarized using the PA scheme with $M = 4$, the binary SSD300 model achieves comparable accuracy by 1.82 mAP reduction compared with its full precision baseline networks. When applying the PA scheme with binary weights ($M = 4$) and binary activations ($N = 5$) for the SSD300 network, the binary SSD300 network shows an accuracy reduction in 15.75 mAP, which outperforms the real-time full-precision Fast YOLO [26] (52.7 mAP).

4.5 Comparisons with state-of-the-art methods

Table 6: Accuracy comparisons of ResNet18 with different quantized methods.

Model	W	A	$Top-1$	$Top-5$
Full Precision	32	32	69.3%	89.2%
BWN	1	32	60.8%	83.0%
XNOR-Net	1	1	51.2%	73.2%
Bi-Real Net	1	1	56.4%	79.5%
ABC-Net ($M = 5$, $N = 5$)	1	1	65.0%	85.9%
Group-Net (5 bases)	1	1	64.8%	85.7%
DoReFa-Net	2	2	62.6%	84.4%
SYQ	1	8	62.9%	84.6%
LQ-Net	2	2	64.9%	85.9%
PA-Net ($M = 8$, $N = 7$)	1	1	68.1%	88.0%
PA-Net ($M = 8$)	1	32	69.3%	88.9%

The comparisons between PA-Net and recent developments are shown in Table 6, where PA-Net adopts the configuration of $M = 8$ and $N = 7$. Regarding single binary models BWN, XNOR-Net [25] and Bi-Real Net [23], and multiple parallel binary models ABC-Net[20] and Group-Net[39], PA-Net outperforms them by much higher accuracy. When it comes to the comparison with fix-point quantization DoreFa-Net [37, 34, 6], fixed-point CNNs can achieve the same or even higher performance with carefully customized bit-widths than PA-Net. But the advantage of PA-Net is the parallelism of inference architecture, which provides a much lower latency using bitwise operation than fixed-point CNNs.

Table 7: Memory usage and Flops calculation of Bi-Real Net, PA-Net, and full precision models.

Model	Memory usage	Memory saving	Flops	Speedup
Bi-Real ResNet18	33.6Mbit	$11.14\times$	1.67×10^8	$10.86\times$
ABC-ResNet18	77.1Mbit	$4.85\times$	6.74×10^8	$2.70\times$
PA-ResNet18	61.6Mbit	$6.08\times$	6.74×10^8	$2.70\times$
ResNet18	374.1Mbit	–	1.81×10^9	–
Bi-Real ResNet34	43.7Mbit	$15.97\times$	1.81×10^8	$18.99\times$
ABC-ResNet34	106.3Mbit	$6.56\times$	1.27×10^9	$2.88\times$
PA-ResNet34	85.0Mbit	$8.20\times$	1.27×10^9	$2.88\times$
ResNet34	697.3Mbit	–	3.66×10^9	–
Bi-Real ResNet50	176.8Mbit	$4.62\times$	5.45×10^8	$7.08\times$
ABC-ResNet50	201.6Mbit	$4.06\times$	1.44×10^9	$2.68\times$
PA-ResNet50	161.3Mbit	$5.07\times$	1.44×10^9	$2.68\times$
ResNet50	817.8Mbit	–	3.86×10^9	–

Table 8: Latency cost of Bi-Real Net, PA-Net, and full precision models. T_{XNOR}, T_{pop}, T_{mul}, T_{AND}, T_{com}, T_{add} refer to the delay time of a XNOR, popcount, multiplication, AND, comparison, and addition operation, respectively.

Model	Latency cost	Speedup
Bi-Real Net	$c_{in}hw\times(T_{XNOR}+T_{pop})+T_{mul}$	$\approx(T_{mul}+T_{add})/(T_{XNOR}+T_{pop})$
PA-Net	$c_{in}hw\times(T_{AND}+T_{pop})+5T_{mul}+4T_{add}+T_{com}$	$\approx(T_{mul}+T_{add})/(T_{AND}+T_{pop})$
Full precision models	$c_{in}hw\times T_{mul}+(c_{in}hw-1)\times T_{add}$	–

4.6 Computational complexity analysis

In this part, we analyze and compare the computational complexity of Bi-Real Net (Liu et al. 2018), PA-Net, and full precision models on current CPUs in terms of computation and memory usage, and on customized hardware (i.e., FPGA/ASIC) in terms of latency. Bi-Real Net maintains high efficiency and achieves the state-of-the-art accuracy as a single binary CNN. During this analysis, PA scheme uses 4 bases for weights and 5 bases for activations approximation.

4.6.1 Computation and memory usage analysis

We analyze and compare the computational complexity of Bi-Real Net [23], PA-Net and full precision models, and their memory saving and speedup are shown in Table 7.

Unlike full precision models which require real-valued parameters and operations, PA-Net and Bi-Real Net have binary and real-valued parameters mixed, so their execution requires both bitwise and real-valued operations. To compute the memory usage of PA-Net and Bi-real Net, we use 32 bit times the number of real-valued parameters and 1 bit times the number of binary values, which are summed together to get their total number bit. We use Flops as the main metrics to measure the bitwise operations, the real-valued operations, and the speedup of implementation. Since the current generation of CPUs can compute bitwise AND and popcount operations in parallelism of 64, the Flops to compute PA-Net and Bi-Real Net is equal to the number of the real-valued multiplications, comparisons, and $1/64$ of the number of the bitwise operations.

We follow the suggestion from [25, 23] to keep the weights and activations of the first convolutional and the last fully connected layer as real-valued. It is worthy to notice that we binarize all the 1×1 downsampling layer in PA-Net to further reduce the computational complexity.

For ResNet18, ResNet34, and ResNet50, our PA scheme can reduce memory usage by more than 5 times and achieves a computation reduction of nearly 3 times, in comparison with the full precision counterpart. Compared with Bi-Real ResNet50, the computation reduction of our proposed PA-ResNet50 with 4 weights bases and 5 activations bases is only two times smaller, and it even requires less memory usage because of the binarization of the downsampling layer.

Combining Table 3 and Table 7, we can conclude that PA-Net can achieve better accuracy (1.6%, 1.7%, and 2.9% for ResNet18, ResNet34, and ResNet50) while consuming fewer parameters (15.4Mbit, 21.3Mbit, and 40.33Mbit for ResNet18, ResNet34, and ResNet50) and the same Flops compared to ABC-Net during the inference stage.

4.6.2 Latency analysis

To be implemented on customized hardware (i.e., FPGA/ASIC), latency cost is one of the most important metrics for real-time applications. As shown in Table 8, the latency cost of an individual convolution in Bi-Real Net, PA-Net, and full precision models is analyzed, where we assume that the convolution implementation is paralleled thoroughly. Compared with full precision models, the latency cost of PA-Net and Bi-Real Net is significantly reduced. T_{AND} is smaller than T_{XNOR}, and the latency cost of a convolution in PA-Net increased only by $4T_{mul}+4T_{add}+T_{com}$ compared with that in Bi-Real Net.

5 Conclusions

In this paper, we introduced the PA scheme for multiple binary CNNs, which adopts piecewise functions for both the forward propagation and backpropagation. Compared with state-of-the-art single and multiple binary CNNs, our scheme provides a better approximation for both full precision weights and activations. We implemented our scheme over several modern CNN architectures, such as ResNet, DenseNet, and MobileNet, and tested on classification task using ImageNet dataset. Results are competitive and almost close the accuracy gap compared with their full precision counterparts. Because of the binarization of downsampling layer, our proposed PA-ResNet50 requires less memory usage and only two times Flops than Bi-Real Net with 4 weights and 5 activations bases, which shows its potential efficiency advantage over single binary CNNs with a deeper network.

REFERENCES

[1] Zhu Baozhou, Nauman Ahmed, Johan Peltenburg, Koen Bertels, and Zaid Al-Ars, 'Diminished-1 fermat number transform for integer convolutional neural networks', in *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*, pp. 47–52. IEEE, (2019).

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, 'Estimating or propagating gradients through stochastic neurons for conditional computation', *arXiv preprint arXiv:1308.3432*, (2013).

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, 'Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1', *arXiv preprint arXiv:1602.02830*, (2016).

[4] Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia, 'Bnn+: Improved binary network training', *arXiv preprint arXiv:1812.11800*, (2018).

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, 'The pascal visual object classes challenge: A retrospective', *International Journal of Computer Vision*, **111**(1), 98–136, (January 2015).

[6] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong, 'Syq: Learning symmetric quantization for efficient deep neural networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4300–4309, (2018).

[7] Joshua Fromm, Shwetak Patel, and Matthai Philipose, 'Heterogeneous bitwidth binarization in convolutional neural networks', in *Advances in Neural Information Processing Systems*, pp. 4006–4015, (2018).

[8] Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen, 'Network sketching: Exploiting binary structure in deep cnns', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5955–5963, (2017).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, (2016).

[10] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang, 'Soft filter pruning for accelerating deep convolutional neural networks', *arXiv preprint arXiv:1808.06866*, (2018).

[11] Zhezhi He and Deliang Fan, 'Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11438–11446, (2019).

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, 'Mobilenets: Efficient convolutional neural networks for mobile vision applications', *arXiv preprint arXiv:1704.04861*, (2017).

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, (2017).

[14] Sergey Ioffe and Christian Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167*, (2015).

[15] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu 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*, pp. 4350–4359, (2019).

[16] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[17] Andrew Lavin and Scott Gray, 'Fast algorithms for convolutional neural networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4013–4021, (2016).

[18] Fengfu Li and Bin Liu, 'Ternary weight networks', *CoRR*, **abs/1605.04711**, (2016).

[19] Zefan Li, Bingbing Ni, Wenjun Zhang, Xiaokang Yang, and Wen Gao, 'Performance guaranteed network acceleration via high-order residual quantization', in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2584–2592, (2017).

[20] Xiaofan Lin, Cong Zhao, and Wei Pan, 'Towards accurate binary convolutional neural network', in *Advances in Neural Information Processing Systems*, pp. 345–353, (2017).

[21] Chunlei Liu, Wenrui Ding, Xin Xia, Baochang Zhang, Jiaxin Gu, Jianzhuang Liu, Rongrong Ji, and David Doermann, 'Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2691–2699, (2019).

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, 'Ssd: Single shot multibox detector', 21–37, (2016).

[23] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng, 'Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm', in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 722–737, (2018).

[24] Asit Mishra and Debbie Marr, 'Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy', *arXiv preprint arXiv:1711.05852*, (2017).

[25] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, 'Xnor-net: Imagenet classification using binary convolutional neural networks', in *European Conference on Computer Vision*, pp. 525–542. Springer, (2016).

[26] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, 'You only look once: Unified, real-time object detection', *CoRR*, **abs/1506.02640**, (2015).

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., 'Imagenet large scale visual recognition challenge', *International Journal of Computer Vision*, **115**(3), 211–252, (2015).

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, 'Mobilenetv2: Inverted residuals and linear bottlenecks', *arXiv preprint arXiv:1801.04381*, (2018).

[29] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang, 'Searching for accurate binary neural architectures', in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, (2019).

[30] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer, 'Efficient processing of deep neural networks: A tutorial and survey', *Proceedings of the IEEE*, **105**(12), 2295–2329, (2017).

[31] Wei Tang, Gang Hua, and Liang Wang, 'How to train a compact binary neural network with high accuracy?', in *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).

[32] Vincent W-S Tseng, Sourav Bhattacharya, Javier Fernández Marqués, Milad Alizadeh, Catherine Tong, and Nicholas D Lane, 'Deterministic binary filters for convolutional neural networks', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2739–2747. AAAI Press, (2018).

[33] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen, 'Tbn: Convolutional neural network with ternary inputs and binary weights', in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 315–332, (2018).

[34] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua, 'Lq-nets: Learned quantization for highly accurate and compact deep neural networks', in *The European Conference on Computer Vision (ECCV)*, (September 2018).

[35] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun, 'Efficient and accurate approximations of nonlinear convolutional networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1984–1992, (2015).

[36] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen, 'Incremental network quantization: Towards lossless cnns with low-precision weights', *arXiv preprint arXiv:1702.03044*, (2017).

[37] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou, 'Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients', *arXiv preprint arXiv:1606.06160*, (2016).

[38] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally, 'Trained ternary quantization', *arXiv preprint arXiv:1612.01064*, (2016).

[39] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid, 'Structured binary neural networks for accurate image classification and semantic segmentation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–422, (2019).