MoBiNet: A Mobile Binary Network for Image Classification

Hai Phan[1]   Dang Huynh[2]   Yihui He[1]   Marios Savvides[1]   Zhiqiang Shen[1]

[1]Carnegie Mellon University   [2]Axon Enterprise

{haithanp,he2,marioss,zhiqians}@andrew.cmu.edu   dhuynh@axon.com

Abstract

*MobileNet and Binary Neural Networks are two among the most widely used techniques to construct deep learning models for performing a variety of tasks on mobile and embedded platforms. In this paper, we present a simple yet efficient scheme to exploit MobileNet binarization at activation function and model weights. However, training a binary network from scratch with separable depth-wise and point-wise convolutions in case of MobileNet is not trivial and prone to divergence. To tackle this training issue, we propose a novel neural network architecture, namely MoBi-Net - Mobile Binary Network in which skip connections are manipulated to prevent information loss and vanishing gradient, thus facilitate the training process. More importantly, while existing binary neural networks often make use of cumbersome backbones such as Alex-Net, ResNet, VGG-16 with float-type pre-trained weights initialization, our MoBi-Net focuses on binarizing the already-compressed neural networks like MobileNet without the need of a pre-trained model to start with. Therefore, our proposal results in an effectively small model while keeping the accuracy comparable to existing ones. Experiments on ImageNet dataset show the potential of the MoBiNet as it achieves **54.40%** top-1 accuracy and dramatically reduces the computational cost with binary operators.*

1. Introduction

The rising of Convolution Neural Networks (CNNs) in Deep Learning has resulted in a variety of significant improvements in complicated tasks such as object detection [7, 42, 32, 56, 55, 46, 47, 54, 16], object segmentation [10, 35, 26, 27], text classification [23, 43], etc. These impressive outcomes have been attributed to the learning capability of the million-parameter structure of convolutional layers in neural networks. Intuitively saying, the complexity of CNNs should be proportional to its capacity and hence, there always exists a trade-off between accuracy and com-

putational cost of a CNN model.

Image classification [12, 25, 48] has attracted many research efforts in recent years. ImageNet [5] stands out to be one of the most popular datasets used for evaluating classification accuracy. This dataset contains millions of images categorized into a wide range of contexts which make the classification task extremely challenging. To cope with this, many models have been proposed: Alex-Net [25], VGG-16 [48], Inception [49], ResNet [12]. These networks are powerful but cumbersome, therefore the deployment on portable and mobile devices is not preferable. To be more specific, Alex-Net and VGG-16 Caffe [21] models are over 200MB and 500MB in size, and take 725M FLOPs and 16 GFLOPs respectively in terms of computational cost. To better suit mobile devices, some techniques have emerged, shedding light on deep compressed neural networks, including model pruning [9, 8, 33, 15], light-weight modules [17, 45, 14], binary network [18, 3, 41, 4] and mimicked network [30, 50].

Building a small but efficient neural network is not trivial. We want to optimize the number of model parameters while preserving comparable accuracy and facilitating ease of training. Pruning methods shrink model size by eliminating insignificant channels that cause redundant computation. However, the search for such channels is expensive and requires to be done in an attentive manner. Light-weight networks such as SqueezeNet [20], MobileNet [17, 45], ShuffleNet [51, 36], ESPNet [44] achieve promising results for not only image classification but object detection and segmentation as well.

Recently applying ideas of AutoML, [13] takes advantage of reinforcement learning to improve the network compression. Additionally, an innovative neural network called Neural Architecture Search (NASNet) [57, 58, 40] examines a database of dimension convolution layers to automate an effective architecture design, surpassing prior human-defined neural networks. However, the search for proper modules is terribly complicated and demands a vast amount of computational resources to serve an exhaustive training

process. Another approach is network binarization which consists of manipulating boolean-type parameters to approximate the deep neural network calculation, so-called Binary Neural Networks (BNNs) [3, 4]. There are several advantages of BNNs. Firstly, thanks to the use of 1-bit representation, the model shrinkage is guaranteed by a proportional factor 32x compared to the traditional 32-bit float type. Secondly, with binary, multiplication of activations and 1-bit weights can be replaced by the bit-wise operation that undoubtedly results in a speed-up. Moreover, xnor and popcount operators are utilized to expedite the neural networks [41]. From the engineering perspective, these operators can be easily implemented on either GPU or CPU-based devices. Finally, even though the binary network's accuracy trade-off is not always negligible in comparison with the float-type counterparts, impressive reduction of computational expense and storage requirement makes deep binary neural network potential for practical applications.

Observing that the binarization of lightweight modules in neural networks can gain considerable outcomes, in this paper we propose *Mobile Binary Network* (or *MoBiNet*), a network that significantly compresses MobileNet [17] to only a few megabytes while preserving good accuracy compared to other similar models [3, 4, 41]. MobileNet feeds data into a depth-wise convolution, then integrates the output with a point-wise convolution, achieving an impressive classification accuracy on the ImageNet dataset [5]. Binarizing the separable depth-wise module is not straightforward due to double-approximation issue: (i) the float-type depth-wise is an imitation of standard float-type convolution behavior and (ii) binary depth-wise is an approximation of the float-type depth-wise. Training a deep neural network consisting of binary depth-wise modules is challenging and prone to divergence due to loss of information and vanishing gradient. To tackle the problem, we introduced skip connections and channel dependency enhancement between separable layers for vanishing gradient avoidance through depth-wise layers. We propose three modules to ease the construction of skip connections: Pre-block, Mid-block, and Post-block where we can observe a convergence guarantee over training epochs. On the other hand, our MoBiNet facilitates the training from scratch, without the need of having a pre-trained model to start with. The architecture is illustrated in Fig. 1.

The main contributions of the paper are as follows:

- We propose a novel compressed architecture called MoBiNet that leverages channel connections and binarizes depth-wise convolution layers. Training from scratch, the MoBiNet with binary separable convolutional layers dramatically improves speed and model size (to only a few megabytes) of neural networks while preserving comparable accuracy to state-of-the-art models.

- We propose the use of skip connections to support the non-trivial network training when binarization happens at both activations and weights of the depth-wise convolution. The MoBiNet exploits this dual binarization on the lightweight module consisting of separable convolutions.

- We propose the use of K-dependency to keep the MoBiNet small and efficient. Instead of increasing the number of convolution channels for better performance (and also larger model size), we control dependency within channels in depth-wise convolution layers.

We experimentally show promising results of MoBiNet to solve image classification in the ImageNet dataset. The result demonstrates the potential of MoBiNet, while being efficiently small, can achieve comparable accuracy and speed as other state-of-the-art large binary networks [41, 34] and even surpass them.

2. Related works

Binarized networks. The weights in deep neural networks are often presented in full-precision or float-type values which requires much memory to store and takes time to deploy convolutional operators in the networks. There are some ideas to convert full-precision networks into binary networks. Courbariaux and Bengio *et al*. [18, 3, 4] proposed an approach to constrain weight values to either -1 or 1, considered as the foundation of binary networks. To be efficient, the binary networks approximate binary values for both weights and activations that bring up comparable performance on traditional datasets: MNIST [28], SVHN [38], and CIFAR-10 [24]. To accelerate the inference process, BinaryNet [41] used xnor and popcount operator to obtain a 7x faster runtime on GPU. However, the BinaryNet does not guarantee a good accuracy on more challenging and diverse dataset like the ImageNet. Rastegari *et al*. [41] proposed Binary Weight Network (BWN) and XNOR-Net which outperform the aforementioned methods on ImageNet by more than 16% in Top-1 accuracy. Nevertheless, BWN only requires weights to be binary while XNOR-Net binarizes the input as well. The scale factors in XNOR-Net help retain the high-level information, even BWN on AlexNet [25] gains 56.8%, slightly better than its full precision (56.6%). Another approach named Bi-Real Net [34] improves BNNs in terms of activations and weight binarization for ResNet [12], using derivative sign function approximation. It also improves weight magnitude related to the gradient, then combines with a pre-trained float-type model and gains impressive accuracy of 56.4% and 62.2% with ResNet-18 and ResNet-34 backbone respectively.

Quantized networks. The general case of binarized networks is that weight values are quantized to a fewer number

(a) Standard convolutional module

(b) MobileNet v1 dw-module [17]

(c) MoBiNet Pre-block

(d) MoBiNet Mid-block
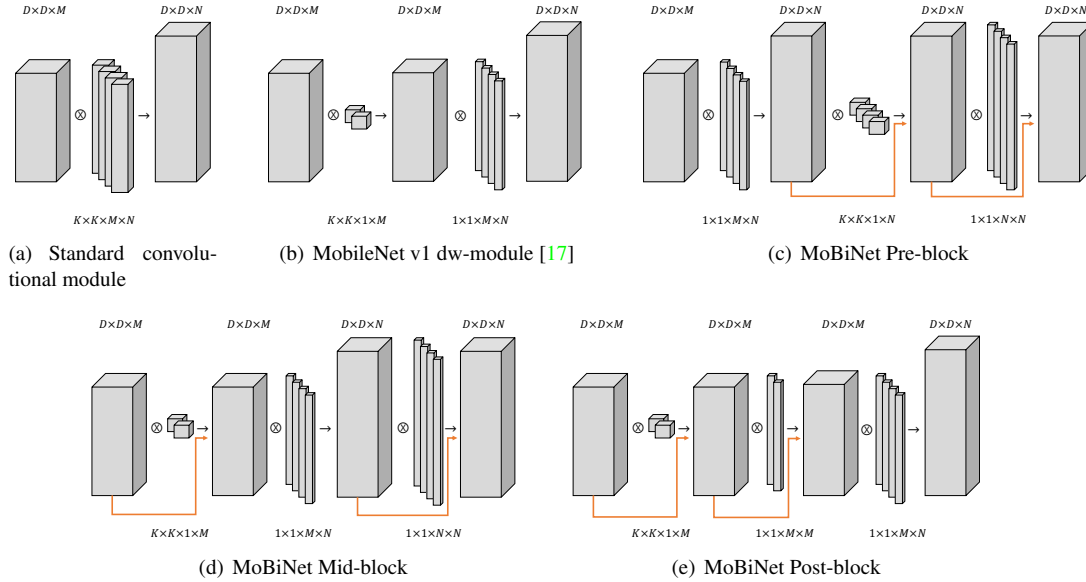
(e) MoBiNet Post-block

Figure 1. Illustration of the standard convolutional layer, depth-wise convolution in MobileNet v1 and the three MoBiNet block designs. The blocks are built with 1×1 binary convolution at different segments: right after the input (Pre-block), between the depth-wise and the point-wise convolution (Mid-block) and right before the output (Post-block). The designs support skip connection creations that allow avoiding loss of information and vanishing gradient when binarization.

of bits. Han *et al*. [8] not only pruned but also quantized unnecessary connections, then fine-tuned remaining weights with less bit representation to gain a speed-up factor. Another approach is to approximate weights as 2^k [1] ($k = -2, -1, 0, 1$, etc.). During the estimation, the network utilizes the shift operator to make computation faster. Ternary Weight Network (TWN) [29] is a variant of the quantized network. It defines zero as a third value for optimizing the cost of the BinaryNet. Zhu and Han *et al*. [53] proposed an incremental approach to train TWN with ternary numbers ($\mathbf{W}_n$, 0, $\mathbf{W}_{np}$) that are learnable. These settings compress the ternary network nearly 16x in comparison with full-precision network and improve the accuracy on ImageNet of precedent ternary networks. Although quantized networks are potential, the deployment on mobile platforms is not trivial and requires much effort.

3. Mobile Binary Network - MoBiNet

In this section, we describe MoBiNet architecture, our proposed novel binary network robust for large scale image classification problem. The depth-wise convolution layers of MobileNet are not suitable to be directly binarized due to accuracy loss and training convergence issue. In MoBiNet, we modify the depth-wise convolutions to make it easier to train, work effectively with both activations and weights binarization, hence accelerate the overall network architecture efficiency. Compared to the XNOR-Net [41], the MoBiNet generalizes the gradient update procedure adjusting the

model weights through back-propagation. The scaling factors for each filter are gradually tuned in an adaptive manner with respect to the gradient update. To further boost the performance of the MoBiNet, K-dependency is exploited to enhance dependency within input and convolution channels. Thanks to this enhancement, the classification accuracy is significantly improved with a minor computational cost trade-off, while the model size is remarkably kept unaffected.

3.1. MoBiNet's Architecture

Float-type separable depth-wise convolution of MobileNet is efficiently accurate and small in size. We build MoBiNet with reuse of this module as it is core of the MobileNet v1 [17] architecture. MoBiNet can be considered as a compact binary version of MobileNet. Specifically, the separable convolution layers contain two lightweight modules: a 3×3 depth-wise convolution with single channel-wise connection and a 1×1 point-wise convolution responsible for linking the depth-wise output channels.

However as mentioned above, the depth-wise convolution binarization can cause severe loss of information, leading to the vanishing gradient issue during training. We notice that skip connections [12] or channels connections can reserve features through many layers, making convolution neural networks converge steadily. From this intuition to ease the skip connection creations, MoBiNet exploits an additional point-wise convolution layer at three positions:
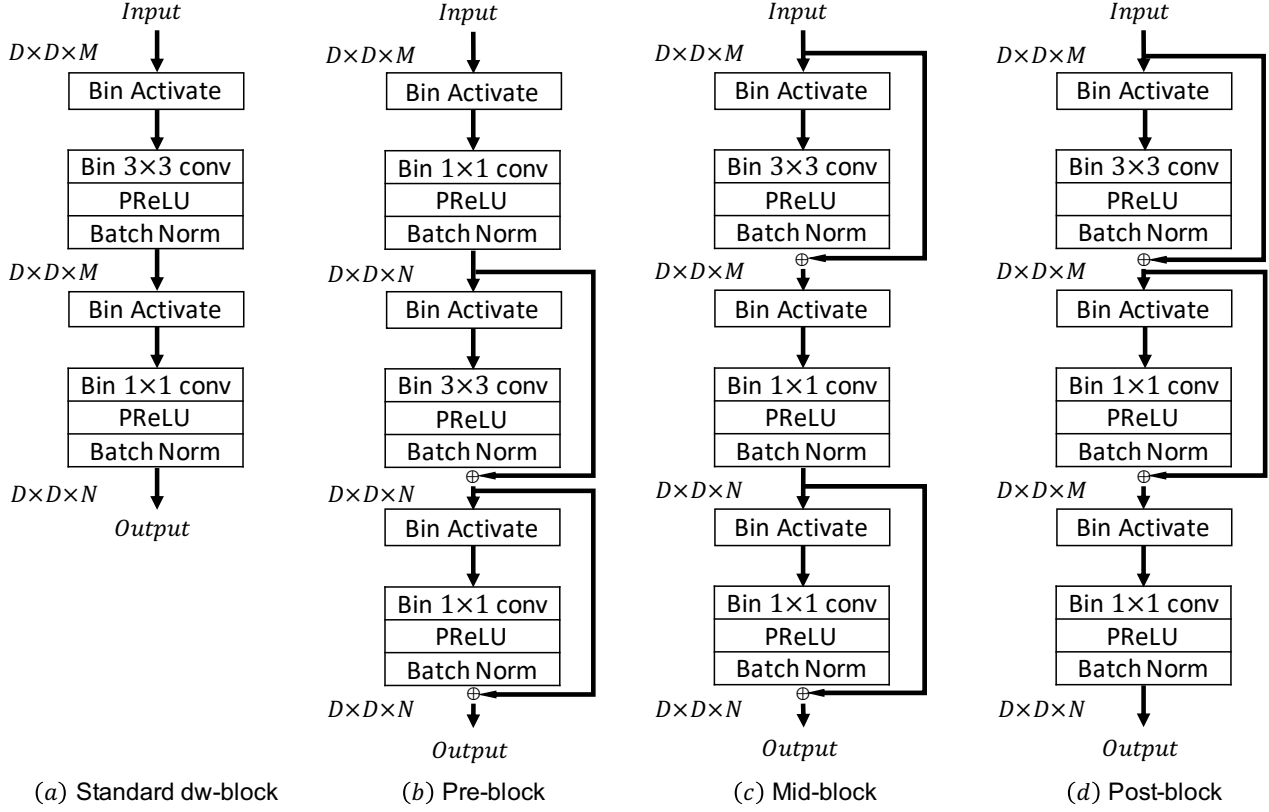
(a) Standard dw-block	(b) Pre-block	(c) Mid-block	(d) Post-block

Figure 2. Detail of (*a*) binary standard separable depth-wise structure in MobileNet v1 and the three binary MoBiNet block structures: (*b*) Pre-block, (*c*) Mid-block and (*d*) Post-block. The binarization happens at both input and weights of every convolutional layer. The convolutions are followed by PReLU and Batch Normalization sequentially.

right after the input (Pre-Block), between depth-wise and point-wise convolution (Mid-Block), and right before the output (Post-Block). The purpose of the block designs is to increase skip connection operators, making separable convolution layers able to keep original features. The architecture detail is illustrated in Fig. 2. We also replace ReLU [37] with PReLU [11] which ensures the convergence stability of MoBiNet. The calculation flow at each convolution segment is as follows: Input → Binary Activation → Binary Convolution → PReLU → Batch Normalization → Output. Unlike MobileNet, to utilize skip connections MoBiNet uses Pooling layers (e.g., max/min/average pooling) to shrink spatial dimension gradually through the network. We opt average pooling for MoBiNet since it experimentally shows a better result than max and min pooling. To prove the impact of skip connections in training MoBiNet, we deploy ablation studies in section 4.2.

3.2. Train Mobile Binary Networks

Binary networks have recently become one of the most prominent approaches in compressed deep neural networks (DNNs) designs. Binary operators can help reduce the com-

putation cost greatly and 1-bit representation of weights supports memory saving which is crucial for mobile devices. To binarize weights and activations, most of BNNs used non-linear $sign$ function to quantize DNNs into -1 and $+1$ as follows:

$$\mathbf{z}^b = \text{sign}(\mathbf{a}) = \begin{cases} +1, & \text{if } \mathbf{a} \geqslant 0, \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{z}^b$ outputs a binary value and $\mathbf{a}$ can be weights or activations. We follow the method described in XNOR-Net [41] to binarize weights and activations in BNNs. However, we modify the way to update parameters to make the training proper. Equation 5 indicates how gradient is updated. Let's denote $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n]$, where $\mathbf{w}_i = \{\mathbf{w}_{ij}\}_{j=1}^m$, $\mathbf{w}_i \in \mathbb{R}^{k \times k \times c}$ is full-precision weight. The activation function for the forward pass of input $\mathbf{x}$ is as follow:

$$\mathbf{a}^i = \sigma(\mathbf{w}^i \mathbf{x}) \quad (2)$$

Let $k \times k$ be filter kernel size, $\mathbf{a}^i$ is an activation output. The most expensive computation in DNNs is the dot-product of inputs and weights $\mathbf{Wx}$. The algorithm con-

verts both weights and activations to 1-bit representation to reduce the required amount of storage memory. It then uses a convolution without multiplication operator to facilitate and accelerate the inference stage. $\mathbf{W}$ is binarized by equation 1 and the optimal solution is $\mathbf{W}^b = \text{sign}(\mathbf{W}) = \{\mathbf{w}_i^b\}_{i=1}^n, \mathbf{w}_i^b \in \{-1, +1\}^{k \times k \times c}$. Similarly, given the input $\mathbf{I} \in \mathbb{R}^{w \times h \times c}$, its binary correspondence is $\mathbf{I}^b = \text{sign}(\mathbf{I}) \in \{-1, +1\}^{w \times h \times c}$. Scaling factor vector is $\boldsymbol{\alpha} \in \mathbb{R}^n$ where $\alpha_i > 0$. The approach estimates scaling factors and binarized both input and weights such that $\mathbf{W} \approx \boldsymbol{\alpha} \cdot \mathbf{W}^b$ and $\mathbf{I} * \mathbf{W} \approx (\mathbf{I}^b \odot \mathbf{W}^b)\boldsymbol{\alpha}$, where $\odot$ indicates xnor-bitcount operator. It can be formulated as an optimization problem:

$$\min_{\mathbf{w}_i^b, \alpha_i} \quad \left\| \mathbf{w}_i - \alpha_i \mathbf{w}_i^b \right\|_2^2$$
$$\text{s.t} \quad \mathbf{w}_i^b \in \{-1, +1\}^{k \times k \times c}, \quad (3)$$
$$\alpha_i > 0.$$

Denote $\hat{\mathbf{w}}_i = \{\alpha_i \cdot \mathbf{w}_{ij}\}_{j=1}^m$. By taking the derivative, the optimal solution for equation 3 is $\mathbf{w}_i^b = \text{sign}(\mathbf{w}_i)$ and $\alpha_i^* = \frac{1}{(\mathbf{w}_i^b)^T \mathbf{w}_i^b} |\mathbf{w}_i|_{l1} = \frac{1}{p} |\mathbf{w}_i|_{l1} = \frac{1}{p} \sum_{j=1}^m |\mathbf{w}_{ij}|$ where $p = k \times k \times c$. It is proved that the optimal binary weights $\mathbf{w}^b$ can be directly computed using a *sign* function and the optimal scaling factors are the average of absolute value of $\mathbf{w}_i$. Let $\hat{\mathbf{W}} = \boldsymbol{\alpha} \cdot \text{sign}(\mathbf{W}) = \{\hat{\mathbf{w}}_i\}_{i=1}^n$ be the approximation of full-precision weights w.r.t binary weights, $\hat{\mathbf{w}}_i = \alpha_i \cdot \text{sign}(\mathbf{w}_i)$. The gradient of the loss function L w.r.t $\mathbf{w}_{ij}$ is computed as follows:

$$\nabla_{\mathbf{w}_{ij}} L = \left(\frac{\partial L}{\partial \mathbf{w}_{ij}} \right)^T \quad (4)$$

$$\frac{\partial L}{\partial \mathbf{w}_{ij}} = \frac{1}{p} \text{sign}(\mathbf{w}_{ij}) \sum_{k=1}^m \frac{\partial L}{\partial \hat{\mathbf{w}}_{ik}} \mathbf{w}_{ik}^b + \frac{\partial L}{\partial \hat{\mathbf{w}}_{ij}} \alpha_i \frac{\partial \mathbf{w}_{ij}^b}{\partial \mathbf{w}_{ij}} \quad (5)$$

In practice when training, full-precision weights are binarized in the forward pass. However in the backward pass, weights are updated with full-precision values. When the training stops, the full-precision weights are completely binarized and saved with 1-bit representation.

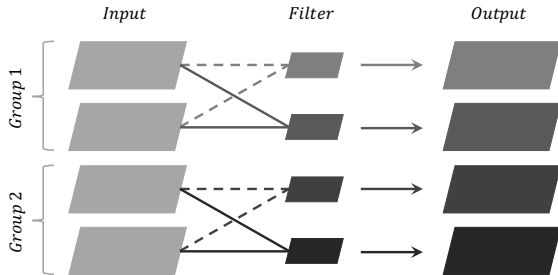### 3.3. Channels Dependency Enhancement

Figure 3. Channel dependency with $c = 4$ and $K = 1$.

In this section, we present how to improve single-layer dependency in depth-wise convolution by exploiting K-layer dependency, or K-dependency for short. The intuition is, due to the lack of channel interaction in single-layer depth-wise, the output suffers severe information loss and it is no longer efficient after being binarized. To cope with this, we provide a more complex dependency between input and convolution channels by defining the term K-dependency. This term allows to control the dependency level of channels and has a flavor of group convolution. Figure 3 depicts how the K-dependency is set up. In the section 4.3, we explore the efficiency of the K-dependency by varying $K = 1, 2, 3$ (i.e., splitting the input and convolution channels into 2^1, 2^2, 2^3 groups) together with Pre-block, Mid-block and Post-block integration. The term K can be also considered as group convolutions with $\#groups = \frac{c}{2^K}$, where c is the number of input channels. If $K = 0$, $\#groups = c$, i.e., the extreme case equivalent to depth-wise convolution. The output activation of a group g can be computed by inner convolution operation within channels and filters in the same group as follows:

$$\mathbf{a}_i^g = \sigma \left(\sum_{j=1}^g \mathbf{w}_i^g \cdot \mathbf{x}_j^g \right) \quad (6)$$

where $\mathbf{a}_i^g$ is the ith output activation at group g, and $\mathbf{w}_i^g, \mathbf{x}_j^g$ is the ith weight and j^{th} input in group g.

4. Experiments and Evaluations

In this section, we analyze the efficiency our proposed MoBiNet. Three ablation studies are conducted. The first is to verify our hypothesis that skip connections are helpful for separable convolution binarization, section 4.2.1 presents comparison between mobile networks with and without skip connection. The second is to evaluate the efficiency of the three extra block design (Pre/Mid/Post-block) added in MoBiNet and compare with the binary original MobileNet. The third is, in section 4.2.3 we present efficiency of replacing ReLU with PReLU for MoBiNet to increase accuracy. In addition, in 4.3 we also demonstrate the impact of K-dependency to the enhancement of MoBiNet, under several selections of hyper-parameter K ($K = 0, 1, 2, 3$). Finally, to indicate that our MoBiNet can achieve comparable outcomes with the other binary neural networks we analyze its result with the state-of-the-art binary neural networks on five metrics: Top-1, Top-5 classification accuracy, number of parameters, and number of FLOPs. Regarding the later two metrics, we also point out numerical gain factors that MoBiNet achieves.

4.1. Dataset and Implementation Details

Dataset. The image classification training and evaluation are carried on ILSVRC12 ImageNet dataset. The ImageNet

is a large scale and diverse dataset containing 1.2M images for training and 50K for testing. They are classified into 1000 categories such as lionfish, airliner, red panda, etc. We opt to use ImageNet because it is such popular that the XNOR-Net [41] and the Bi-RealNet also use for benchmarking purpose, thus making the comparison with MoBiNet fair. Each image in ImageNet is scaled to the size of 256×256 pixels. For training, images are randomly cropped to fit 224×224. For testing, we apply center crop with the same size of 224×224.

Implementation details. All experiments are implemented using PyTorch deep learning framework [39]. We run tests with Pre-block, Mid-block, and Post-block designs for MoBiNet. During the training of binary neural networks, we process the float-type model and the binary model concurrently. That means the floating-point weights are stored in RAM for weight update through backpropagation and they are binarized to compute activation in the next layer. For every experiment, we train the model in 50 epochs and learning rates are set to 10^{-3} to train the Pre-block and 10^{-4} for training the Mid-block and Post-block. We choose the mini batch size 128, weight decay 10^{-5} and momentum 0.9 with Adam optimizer [22]. At some points, the convergence becomes slow and requires manual learning rate update. To foster the training, the three blocks receive the same multiplicative decrease factor 0.1 at different points of time: the Mid-block is at 20th and 30th epoch whereas the Pre-block and the Post-block are at 15th and 45th epoch.

In the first convolution layer, feature representation often has a low depth dimension (e.g., 3 for visual RGB image). To avoid the loss of crucial information, the first convolution layer is not binarized. Also, the output features extracted by MoBiNet are not binarized either to increase the capability of classification. These make the binary model compression factor slightly less than 32x in the context of 32-bit float type and binary type. For the rest of the network, both input and weights are binarized, then multiplied with the according scaling factors α_i. Weights are updated following Equation 5. When the training finished, the models are saved with 1-bit representation to reduce memory storage for inference.

4.2. Ablation studies

4.2.1 The Efficiency of Skip Connections In MoBiNet

Recall that training binary separable convolution is prone to unstable convergence due to considerable loss of information through layers. The fact that training loss diverges indicates a serious issue in the mobile binary neural network. To overcome this obstacle, we deploy skip connections between separable convolution layers to aid the training. In this experiment, we test the efficiency of the skip connections in MoBiNet. The skip connection performance of Pre-
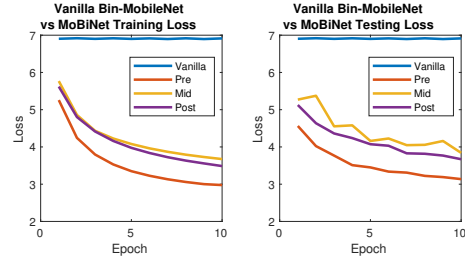

Figure 4. Convergence of Vanilla Bin-MobileNet vs MoBiNet.

block, Mid-block and Post-block is reported for training and test phase. We also present the loss values of without-skip-connection in vanilla binary Mobilenet through epochs in Fig. 4 to show that training does not converge. The per-
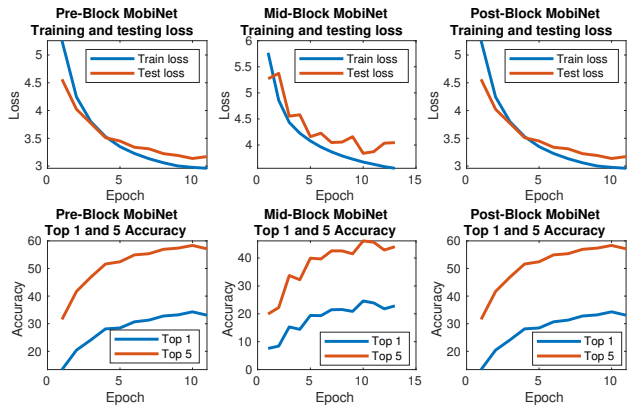

Figure 5. Training and testing performance (with the loss (top) and accuracy (bottom)) when using skip connections in the three block designs.

formance of training with skip connections is shown in Figure 5 with the three blocks. We pick $K = 0$, i.e., zero-dependency and equivalent to the original depth-wise convolution. We stopped the training at 10th and 15th epoch to see the effect. Even though the convergence slope and accuracy are not decent due to the zero-dependency, one can observe that the training goes steady over time. It confirms that our hypothesis of using skip connections for separable convolution binarization can deal with the divergence issue by limiting the loss of information. Without loss of generality, it is applicable to binarize weights of not only in MoBiNet but other neural networks [44, 51, 36, 2] using depth-wise convolution as well.

4.2.2 Efficiency of MoBiNet modules: Pre-block, Mid-block and Post-block

Skip connections clearly help mobile neural network with separable convolution layers converge and achieve reasonable results. To further increase the efficiency of the skip

Networks	Top-1 Accuracy (%)	Top-5 Accuracy (%)
W/o blocks	25.46	47.34
Pre-block	**35.86**	**59.46**

Table 1. Top-1 and Top-5 accuracy (in percentage) comparison between MoBiNet with and without the Pre-block design
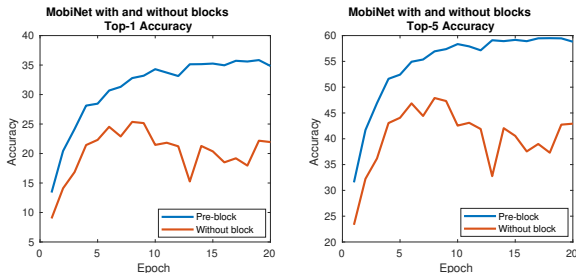

Figure 6. Training performance: Top-1 and Top-5 accuracy of MoBiNet with and without Pre-block through 20 first epochs.

connection and explore its potential, we proposed an incremental 1×1 point-wise convolution layer to help upsample the features and support the skip connection creation. The 1×1 point-wise convolution layer is placed at three different positions: right after the input (Pre-block), between depthwise and point-wise convolution (Mid-block), and right before the output (Post-block). Figure 2 illustrates the locations of the 1×1 convolution. In this evaluation, we pick the Pre-block design as a representative to conduct experiments in two scenarios: with and without the block design, then analyze the impact. The authors also observe a consistent result in the case of Mid-block and Post-block.

The experiments are set up as follows:

- We construct MoBiNet with and without Pre-block with zero-dependency depth-wise convolution layers (i.e., choosing $K = 0$).

- The MoBiNet is built with aforementioned hyperparameters as in section 4.1.

- When training MoBiNet with and without Pre-block design, we run in 20 epochs, starting with an initial learning rate of 10^{-4} and updating the rate at the 15th epoch with a decay factor 0.1. The input and weights are both binarized except the first convolutional layer and the classifier layer right before the classification output.

- We report Top-1 and Top-5 accuracy for comparison, then plot the training curve through epochs.

The results are shown in Table 1 and training performance is in Figure 6. Convergence of MoBiNet with the blocks is more stable and gains better accuracy while the without-block design fluctuates as the training goes on. It

Networks	ReLU	PReLU
Mid-block ($K = 0$)	32.44	**34.73**
Mid-block ($K = 1$)	48.52	**50.41**
Mid-block ($K = 2$)	49.60	**52.25**
Mid-block ($K = 3$)	51.36	**53.47**

Table 2. Benefit of using PReLU instead of ReLU. The accuracy (in percentage) is reported with Top-1 accuracy on ImageNet. The use of PReLU for MoBiNet improves the accuracy approximately 2% in comparison with ReLU.

confirms that an extra 1×1 point-wise layer for MoBiNet's block design can ease the training of the separable convolution layer.

4.2.3 Efficiency of PReLU layer used in MoBiNet

To further ameliorate the MoBiNet, we replace ReLU with PReLU as it experimentally shows a better feature representation throughout the binary network. In addition, PReLU layers help MoBiNet increase accuracy without incurring computational cost or parameter storage. The efficiency of PReLU is presented in Table 2. We trained the Mid-block designe with $K = 0, 1, 2, 3$ where settings of ReLu and PReLU are exactly the same.

4.3. K-Dependency for MoBiNet Enhancement

The separable convolution layers are limited for feature representation because there is no interaction between layers. To improve this capability for MoBiNet, we proposed a novel method using K-dependency to augment correlation within separate channels. In this section, we show that the factor K plays a key role to boost the accuracy of MoBi-Net. The larger the K, the deeper the channels correlate and hence the more adequately the feature is represented after being convolved. Our selections for the level of dependency are $K = 0, 1, 2, 3$. We report Top-1 and Top-5 accuracy on ImageNet data. The number of FLOPs is computed in a similar way mentioned in [34] to ensure the comparison fairness with the other binary networks. The results are reported in Table 3. Generally speaking, the MoBiNet speeds up the inference stage by $11\times$ compared to MobileNet [17]. The best performance we can achieve is with Mid-block design: Top-1 accuracy of **53.47**% with $K = 3$ and speed-up factor **12**$\times$. In Table 4, we show the efficiency of MoBiNet against several recent models: XNOR-Net [41], Bi-RealNet [34], ABC-Net [31], DoReFa-Net [52], SQ-BWN [6], etc. For efficiency comparison, we opt for Bi-RealNet-18 as a baseline because this approach is the most recent and efficient binary approach. Our MoBi-Net, with Mid-block design and $K = 4$, achieves a speed-up factor of **3.13**$\times$ and a model saving factor of **1.34**$\times$. Furthermore, in terms of accuracy MoBiNet outperforms all of

Networks	K	Top-1 accuracy(%)	Top-5 accuracy(%)	Mem usage (MB)	FLOPs (M)	Speed up
Pre-block	0	35.86	59.46	4.60	45.87	12.40×
	1	48.93	72.83		46.57	12.23×
	2	50.74	74.22		47.97	11.86×
	3	51.41	74.75		50.76	11.21×
Mid-block	0	34.73	58.28	4.60	45.61	12.40×
	1	50.41	73.98		46.04	12.48×
	2	52.25	75.32		46.90	12.36×
	3	**53.47**	**76.46**		48.62	12.13×
Post-block	0	33.58	56.69	4.50	35.37	16.09×
	1	46.83	70.57		35.80	15.89×
	2	48.22	71.97		36.67	15.52×
	3	49.88	73.27		38.39	14.82×
Mobile Net [17]	Full	70.9	89.90	16.20	569	1.00×

Table 3. The K-dependency performance of MoBiNet including Top-1 and Top-5 accuracy in percentage, memory usage in Megabytes (Pytorch model saving [39]), and FLOPs computed as in [34]. The comparison with full-precision MobileNet regards accuracy and inference speed-up factor. Among the three block designs, the Mid-block yields better accuracy than Pre-block and Post-block. This is because the Mid-block exploits skip connections to preserve both original and scaled features, unlike Pre-block (only scaled feature) and Post-block (only original feature).

Networks	Top-1 accuracy(%)	Top-5 accuracy(%)	#Params (M)	Saving	FLOPs	Speed up
Binary Connect [4]	35.40	61.00	-	-	-	-
BNNs [19]	42.20	67.10	-	-	-	-
ABC-Net [31]	42.70	67.60	-	-	-	-
DoReFa-Net [52]	43.60	-	-	-	-	-
BWN [6]	39.20	64.70	-	-	-	-
SQ-BWN [6]	45.50	70.60.07	-	-	-	-
XNOR-AlexNet [41]	44.20	69.20	62.38	0.19×	1.38×10^8	1.18×
XNOR-ResNet-18 [41]	51.20	69.30	11.7	1.00×	1.67×10^8	0.97×
Bi-RealNet-18 [34]	56.40	79.50	11.70	1.00×	1.63×10^8	1.00×
MoBiNet-Mid ($K = 3$)	**53.47**	**76.46**	**8.50**	**1.38×**	$\mathbf{0.49 \times 10^8}$	**3.33×**
MoBiNet-Mid ($K = 4$)	**54.40**	**77.50**	**8.75**	**1.34×**	$\mathbf{0.52 \times 10^8}$	**3.13×**

Table 4. The Top-1 and Top-5 accuracy comparison between MoBiNet Mid-Block and the state-of-the-art. The Bi-RealNet-18 is selected as a baseline model to make comparisons.

the precedent methods except Bi-RealNet that we are 2% behind. However on the ImageNet dataset, to reproduce the accuracy of the Bi-RealNet, a pretrained model is required whereas the MoBiNet has the freedom to start the training from scratch. Furthermore, one can bridge the accuracy gap by increasing the factor $K > 4$ with an acceptable FLOPs trade-off.

5. Conclusion

We introduced MoBiNet, the very first work for lightweight module binarization. We explore the training efficiency of the binarization with the support of skip connection, the three block designs and the K-dependency. The construction of MoBiNet facilitates the non-trivial training of a binary network, and without loss of generality. MoBiNet can be used for more challenge task: object detection.

References

[1] Y. G. L. X. Y. C. Aojun Zhou, Anbang Yao. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations,ICLR2017*, 2017.

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1800–1807. IEEE Computer Society, 2017.

[3] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[4] M. Courbariaux, Y. Bengio, and J. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[5] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[6] Y. Dong, R. Ni, J. Li, Y. Chen, H. Su, and J. Zhu. Stochastic quantization for learning accurate low-bit deep neural networks. *International Journal of Computer Vision*, Mar 2019.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014.

[8] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[9] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[14] Y. He, X. Liu, H. Zhong, and Y. Ma. Addressnet: Shift-based primitives for efficient convolutional neural networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1213–1222. IEEE, 2019.

[15] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[16] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019.

[17] 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. *CoRR*, abs/1704.04861, 2017.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

[19] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc., 2016.

[20] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size. *CoRR*, abs/1602.07360, 2017.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678. ACM, 2014.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[23] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes. Rmdl: Random multimodel deep learning for classification. In *Proceedings of the 2Nd International Conference on Information System and Data Mining*, ICISDM '18, pages 19–28, New York, NY, USA, 2018. ACM.

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[26] H. N. Le, C. N. Duong, K. Luu, and M. Savvides. Deep contextual recurrent residual networks for scene labeling. In *Journal of Pattern Recognition*, 2018.

[27] H. N. Le, K. G. Quach, K. Luu, and M. Savvides. Reformulating level sets as deep recurrent neural network approach to semantic segmentation. In *Trans. on Image Processing (TIP)*, 2018.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

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

[30] Q. Li, S. Jin, and J. Yan. Mimicking very efficient network for object detection. *2017 IEEE Conference on CVPR*, pages 7341–7349, 2017.

[31] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 345–353. Curran Associates, Inc., 2017.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV (1)*, volume 9905 of *Lecture Notes in Computer Science*, pages 21–37, 2016.

[33] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763, 2017.

[34] 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 *ECCV*, 2018.

[35] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[36] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[37] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010.

[38] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[39] 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. 2017.

[40] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4095–4104, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[41] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2016.

[42] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[43] D. S. Sachan and Petuum. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *AAAI 2019*, 2019.

[44] A. C. L. S. Sachin Mehta, Mohammad Rastegari and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018.

[45] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[46] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[47] Z. Shen, H. Shi, J. Yu, H. Phan, R. S. Feris, L. Cao, D. Liu, X. Wang, T. S. Huang, and M. Savvides. Improving object detection from scratch via gated feature reuse. In *BMVC*, 2019.

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[50] Y. Wei, X. Pan, H. Qin, and J. Yan. Quantization mimic: Towards very tiny cnn for object detection. *CoRR*, abs/1805.02152, 2018.

[51] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[52] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefanet: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

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

[54] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[55] C. Zhu, Y. Ran, K. Luu, and M. Savvides. Seeing small faces from robust anchor's perspective. In *CVPR*, 2018.

[56] C. Zhu, Y. Zheng, K. Luu, and M. Savvides. Enhancing interior and exterior deep facial features for face detection in the wild. In *International Conference on Automatic Face and Gesture Recognition (FG)*, 2018.

[57] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. 2017.

[58] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. 2018.