Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm

Zechun Liu[1] , Baoyuan Wu[2] , Wenhan Luo[2], Xin Yang[3]⋆, Wei Liu[2], and
Kwang-Ting Cheng[1]

[1] Hong Kong University of Science and Technology
[2] Tencent AI lab
[3] Huazhong University of Science and Technology
`zliubq@connect.ust.hk`, {`wubaoyuan1987, whluo.china`}`@gmail.com`,
`xinyang2014@hust.edu.cn, wliu@ee.columbia.edu, timcheng@ust.hk`

Abstract. In this work, we study the 1-bit convolutional neural networks (CNNs), of which both the weights and activations are binary. While being efficient, the classification accuracy of the current 1-bit CNNs is much worse compared to their counterpart real-valued CNN models on the large-scale dataset, like ImageNet. To minimize the performance gap between the 1-bit and real-valued CNN models, we propose a novel model, dubbed Bi-Real net, which connects the real activations (after the 1-bit convolution and/or BatchNorm layer, before the sign function) to activations of the consecutive block, through an identity shortcut. Consequently, compared to the standard 1-bit CNN, the representational capability of the Bi-Real net is significantly enhanced and the additional cost on computation is negligible. Moreover, we develop a specific training algorithm including three technical novelties for 1-bit CNNs. Firstly, we derive a tight approximation to the derivative of the non-differentiable sign function with respect to activation. Secondly, we propose a magnitude-aware gradient with respect to the weight for updating the weight parameters. Thirdly, we pre-train the real-valued CNN model with a clip function, rather than the ReLU function, to better initialize the Bi-Real net. Experiments on ImageNet show that the Bi-Real net with the proposed training algorithm achieves 56.4% and 62.2% top-1 accuracy with 18 layers and 34 layers, respectively. Compared to the state-of-the-arts (*e.g.*, XNOR Net), Bi-Real net achieves up to 10% higher top-1 accuracy with more memory saving and lower computational cost.

1   Introduction

Deep Convolutional Neural Networks (CNNs) have achieved substantial advances in a wide range of vision tasks, such as object detection and recognition [12, 23, 25, 5, 3, 20], depth perception [2, 16], visual relation detection [29, 30], face

---

⋆ Corresponding author.

tracking and alignment [24, 32, 34, 28, 27], object tracking [17], etc. However, the superior performance of CNNs usually requires powerful hardware with abundant computing and memory resources. For example, high-end Graphics Processing Units (GPUs). Meanwhile, there are growing demands to run vision tasks, such as augmented reality and intelligent navigation, on mobile hand-held devices and small drones. Most mobile devices are not equipped with a powerful GPU neither an adequate amount of memory to run and store the expensive CNN model. Consequently, the high demand for computation and memory becomes the bottleneck of deploying the powerful CNNs on most mobile devices. In general, there are three major approaches to alleviate this limitation. The first is to reduce the number of weights, such as Sparse CNN [15]. The second is to quantize the weights (*e.g.*, QNN [8] and DoReFa Net [33]). The third is to quantize both weights and activations, with the extreme case of both weights and activations being binary.

In this work, we study the extreme case of the third approach, *i.e.*, the binary CNNs. It is also called 1-bit CNNs, as each weight parameter and activation can be represented by 1-bit. As demonstrated in [19], up to $32\times$ memory saving and $58\times$ speedup on CPUs have been achieved for a 1-bit convolution layer, in which the computationally heavy matrix multiplication operations become light-weighted bitwise XNOR operations and bit-count operations. The current binarization method achieves comparable accuracy to real-valued networks on small datasets (*e.g.*, CIFAR-10 and MNIST). However on the large-scale datasets (*e.g.*, ImageNet), the binarization method based on AlexNet in [7] encounters severe accuracy degradation, *i.e.*, from 56.6% to 27.9% [19]. It reveals that the capability of conventional 1-bit CNNs is not sufficient to cover great diversity in large-scale datasets like ImageNet. Another binary network called XNOR-Net [19] was proposed to enhance the performance of 1-bit CNNs, by utilizing the absolute mean of weights and activations.

The objective of this study is to further improve 1-bit CNNs, as we believe its potential has not been fully explored. One important observation is that during the inference process, 1-bit convolution layer generates integer outputs, due to the bit-count operations. The integer outputs will become real values if there is a BatchNorm [10] layer. But these real-valued activations are then binarized to -1 or $+1$ through the consecutive sign function, as shown in Fig. 1(a). Obviously, compared to binary activations, these integers or real activations contain more information, which is lost in the conventional 1-bit CNNs [7]. Inspired by this observation, we propose to keep these real activations via adding a simple yet effective shortcut, dubbed Bi-Real net. As shown in Fig. 1(b), the shortcut connects the real activations to an addition operator with the real-valued activations of the next block. By doing so, the representational capability of the proposed model is much higher than that of the original 1-bit CNNs, with only a negligible computational cost incurred by the extra element-wise addition and without any additional memory cost.

Moreover, we further propose a novel training algorithm for 1-bit CNNs including three technical novelties:
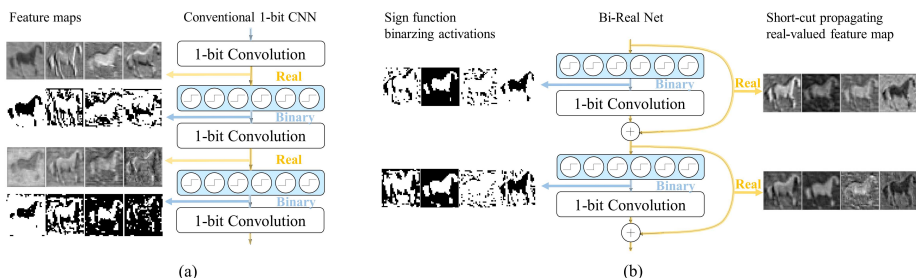
Fig. 1. Network with intermediate feature visualization, yellow lines denote value propagated inside the path being real while blue lines denote binary values. (a) 1-bit CNN without shortcut (b) proposed Bi-Real net with shortcut propagating the real-valued features.

– **Approximation to the derivative of the sign function with respect to activations.** As the sign function binarizing the activation is non-differentiable, we propose to approximate its derivative by a piecewise linear function in the backward pass, derived from the piecewise polynomial function that is a second-order approximation of the sign function. In contrast, the approximated derivative using a step function (*i.e.*, $1_{|x|<1}$) proposed in [7] is derived from the clip function (*i.e.*, clip(-1,x,1)), which is also an approximation to the sign function. We show that the piecewise polynomial function is a closer approximation to the sign function than the clip function. Hence, its derivative is more effective than the derivative of the clip function.

– **Magnitude-aware gradient with respect to weights.** As the gradient of loss with respect to the binary weight is not large enough to change the sign of the binary weight, the binary weight cannot be directly updated using the standard gradient descent algorithm. In BinaryNet [7], the real-valued weight is first updated using gradient descent, and the new binary weight is then obtained through taking the sign of the updated real weight. However, we find that the gradient with respect to the real weight is only related to the sign of the current real weight, while independent of its magnitude. To derive a more effective gradient, we propose to use a magnitude-aware sign function during training, then the gradient with respect to the real weight depends on both the sign and the magnitude of the current real weight. After convergence, the binary weight (*i.e.*, -1 or +1) is obtained through the sign function of the final real weight for inference.

– **Initialization.** As a highly non-convex optimization problem, the training of 1-bit CNNs is likely to be sensitive to initialization. In [17], the 1-bit CNN model is initialized using the real-valued CNN model with the ReLU function pre-trained on ImageNet. We propose to replace ReLU by the clip function in pre-training, as the activation of the clip function is closer to the binary activation than that of ReLU.

Experiments on ImageNet show that the above three ideas are useful to train 1-bit CNNs, including both Bi-Real net and other network structures. Specifically, their respective contributions to the improvements of top-1 accuracy are up to 12%, 23% and 13% for a 18-layer Bi-Real net. With the dedicatedly-designed shortcut and the proposed optimization techniques, our Bi-Real net, with only binary weights and activations inside each 1-bit convolution layer, achieves 56.4% and 62.2% top-1 accuracy with 18-layer and 34-layer structures, respectively, with up to 16.0× memory saving and 19.0× computational cost reduction compared to the full-precision CNN. Comparing to the state-of-the-art model (*e.g.*, XNOR-Net), Bi-Real net achieves 10% higher top-1 accuracy on the 18-layer network.

2   Related Work

Reducing the number of parameters. Several methods have been proposed to compress neural networks by reducing the number of parameters and neural connections. For instance, He et al. [5] proposed a bottleneck structure which consists of three convolution layers of filter size 1×1, 3×3 and 1×1 with a short-cut connection as a preliminary building block to reduce the number of parameters and to speed up training. In SqueezeNet [9], some 3×3 convolutions are replaced with 1×1 convolutions, resulting in a 50× reduction in the number of parameters. FitNets [21] imitates the soft output of a large teacher network using a thin and deep student network, and in turn yields 10.4× fewer parameters and similar accuracy to a large teacher network on the CIFAR-10 dataset. In Sparse CNN [15], a sparse matrix multiplication operation is employed to zero out more than 90% of parameters to accelerate the learning process. Motivated by the Sparse CNN, Han et al. proposed Deep Compression [4] which employs connection pruning, quantization with retraining and Huffman coding to reduce the number of neural connections, thus, in turn, reduces the memory usage.

Parameter quantization. The previous study [13] demonstrated that real-valued deep neural networks such as AlexNet [12], GoogLeNet [25] and VGG-16 [23] only encounter marginal accuracy degradation when quantizing 32-bit parameters to 8-bit. In Incremental Network Quantization, Zhou et al. [31] quantize the parameter incrementally and show that it is even possible to further reduce the weight precision to 2-5 bits with slightly higher accuracy than a full-precision network on the ImageNet dataset. In BinaryConnect [1], Courbariaux et al. employ 1-bit precision weights (1 and -1) while maintaining sufficiently high accuracy on the MNIST, CIFAR10 and SVHN datasets.

Quantizing weights properly can achieve considerable memory savings with little accuracy degradation. However, acceleration via weight quantization is limited due to the real-valued activations (*i.e.*, the input to convolution layers).

Several recent studies have been conducted to explore new network structures and/or training techniques for quantizing both weights and activations while minimizing accuracy degradation. Successful attempts include DoReFa-Net [33] and QNN [8], which explore neural networks trained with 1-bit weights
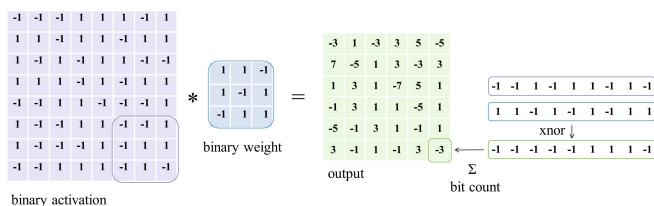
Fig. 2. The mechanism of xnor operation and bit-counting inside the 1-bit CNNs presented in [19].

and 2-bit activations, and the accuracy drops by 6.1% and 4.9% respectively on the ImageNet dataset compared to the real-valued AlexNet. Additionally, BinaryNet [7] uses only 1-bit weights and 1-bit activations in a neural network and achieves comparable accuracy as full-precision neural networks on the MNIST and CIFAR-10 datasets. In XNOR-Net [19], Rastegari et al. further improve BinaryNet by multiplying the absolute mean of the weight filter and activation with the 1-bit weight and activation to improve the accuracy. ABC-Net [14] proposes to enhance the accuracy by using more weight bases and activation bases. The results of these studies are encouraging, but admittedly, due to the loss of precision in weights and activations, the number of filters in the network (thus the algorithm complexity) grows in order to maintain high accuracy, which offsets the memory saving and speedup of binarizing the network.

In this study, we aim to design 1-bit CNNs aided with a real-valued shortcut to compensate for the accuracy loss of binarization. Optimization strategies for overcoming the gradient dismatch problem and discrete optimization difficulties in 1-bit CNNs, along with a customized initialization method, are proposed to fully explore the potential of 1-bit CNNs with its limited resolution.

3   Methodology

3.1   Standard 1-bit CNNs and Its Representational Capability

1-bit convolutional neural networks (CNNs) refer to the CNN models with binary weight parameters and binary activations in intermediate convolution layers. Specifically, the binary activation and weight are obtained through a sign function,

$$a_b = \text{Sign}(a_r) = \begin{cases} -1 \text{ if } \ a_r < 0 \\ +1 \text{ otherwise} \end{cases}, \qquad w_b = \text{Sign}(w_r) = \begin{cases} -1 \text{ if } w_r < 0 \\ +1 \text{ otherwise} \end{cases}, \quad (1)$$

where a_r and w_r indicate the real activation and the real weight, respectively. a_r exists in both training and inference process of the 1-bit CNN, due to the convolution and batch normalization (if used). As shown in Fig. 2, given a binary activation map and a binary 3×3 weight kernel, the output activation could be

Map size | Value range | Different value | ℝ

(a)

	Map size	Value range	Different value	ℝ
Sign				
A_b^l	14x14x32	{-1, 1}	2	$2^{14 \times 14 \times 32}$
1-bit Conv				
A_m^l	14x14x32	{-288,-286,...,288}	289	$289^{14 \times 14 \times 32}$
BatchNorm				
A_r^{l+1}	14x14x32	{a certain range}	289	$289^{14 \times 14 \times 32}$
Sign				
A_b^{l+1}	14x14x32	{-1, 1}	2	$2^{14 \times 14 \times 32}$

(b)

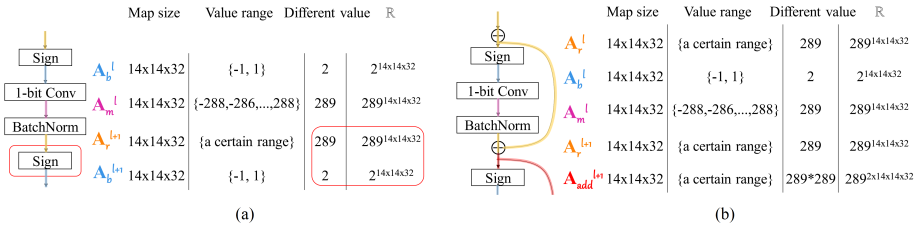|	Map size	Value range	Different value	ℝ
Sign				
A_r^l	14x14x32	{a certain range}	289	$289^{14 \times 14 \times 32}$
A_b^l	14x14x32	{-1, 1}	2	$2^{14 \times 14 \times 32}$
1-bit Conv				
A_m^l	14x14x32	{-288,-286,...,288}	289	$289^{14 \times 14 \times 32}$
BatchNorm				
A_r^{l+1}	14x14x32	{a certain range}	289	$289^{14 \times 14 \times 32}$
A_{add}^{l+1}	14x14x32	{a certain range}	289*289	$289^{2 \times 14 \times 14 \times 32}$

Fig. 3. The representational capability ($\mathbb{R}$) of each layer in (a) 1-bit CNNs without shortcut (b) 1-bit CNNs with shortcut. $\mathbf{A}_b^l$ indicates the output of the Sign function; $\mathbf{A}_m^l$ denotes the output of the 1-bit convolution layer; $\mathbf{A}_r^{l+1}$ represents the output of the BatchNorm layer; The superscript l indicates the block index.

any odd integer from -9 to 9. If a batch normalization is followed, as shown in Fig. 3, then the integer activation will be transformed into real values. The real weight will be used to update the binary weights in the training process, which will be introduced later.

Compared to the real-valued CNN model with the 32-bit weight parameter, the 1-bit CNNs obtains up to $32\times$ memory saving. Moreover, as the activation is also binary, then the convolution operation could be implemented by the bitwise XNOR operation and a bit-count operation[19]. One simple example of the bitwise operation is shown in Fig. 2. In contrast, the convolution operation in real-valued CNNs is implemented by the expensive real value multiplication. Consequently, the 1-bit CNNs could obtain up to $64\times$ computation saving.

However, it has been demonstrated in [7] that the classification performance of the 1-bit CNNs is much worse than that of the real-valued CNN models on large-scale datasets like ImageNet. We believe that the poor performance of 1-bit CNNs is caused by its low representational capacity. We denote $\mathbb{R}(\mathbf{x})$ as the representational capability of $\mathbf{x}$, *i.e.*, the number of all possible configurations of $\mathbf{x}$, where $\mathbf{x}$ could be a scalar, vector, matrix or tensor. For example, the representational capability of 32 channels of a binary 14×14 feature map $\mathbf{A}$ is $\mathbb{R}(\mathbf{A}) = 2^{14 \times 14 \times 32} = 2^{6272}$. Given a $3 \times 3 \times 32$ binary weight kernel $\mathbf{W}$, each entry of $\mathbf{A} \otimes \mathbf{W}$ (*i.e.*, the bitwise convolution output) can choose the even values from (-288 to 288), as shown in Fig 3. Thus, $\mathbb{R}(\mathbf{A} \otimes \mathbf{W}) = 289^{6272}$. Note that since the BatchNorm layer is a unique mapping, it will not increase the number of different choices but scale the (-288,288) to a particular value. If adding the 1-bit convolution layer behind the output, each entry in the feature map is binarized, and the representational capability shrinks to 2^{6272} again.

3.2   Bi-Real Net Model and Its Representational Capability

We propose to preserve the real activations before the sign function to increase the representational capability of the 1-bit CNN, through a simple shortcut. Specifically, as shown in Fig. 3(b), one block indicates the structure that "Sign $\rightarrow$ 1-bit convolution $\rightarrow$ batch normalization $\rightarrow$ addition operator". The shortcut
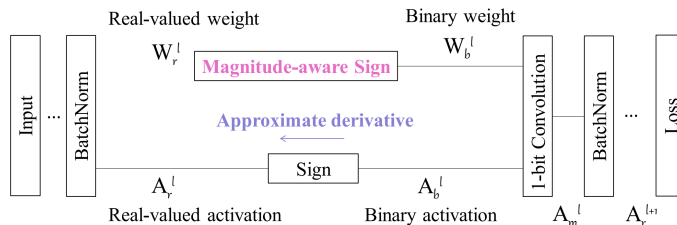
Fig. 4. A graphical illustration of the training process of the 1-bit CNNs, with A being the activation, W being the weight, and the superscript l denoting the lth block consisting with Sign, 1-bit Convolution, and BatchNorm. The subscript r denotes real value, b denotes binary value, and m denotes the intermediate output before the BatchNorm layer.

connects the input activations to the sign function in the current block to the output activations after the batch normalization in the same block, and these two activations are added through an addition operator, and then the combined activations are inputted to the sign function in the next block. The representational capability of each entry in the added activations is 289^2. Consequently, the representational capability of each block in the 1-bit CNN with the above shortcut becomes $(289^2)^{6272}$. As both real and binary activations are kept, we call the proposed model as Bi-Real net.

The representational capability of each block in the 1-bit CNN is significantly enhanced due to the simple identity shortcut. The only additional cost of computation is the addition operation of two real activations, as these real activations already exist in the standard 1-bit CNN (*i.e.*, without shortcuts). Moreover, as the activations are computed on the fly, no additional memory is needed.

3.3   Training Bi-Real Net

As both activations and weight parameters are binary, the continuous optimization method, *i.e.*, the stochastic gradient descent(SGD), cannot be directly adopted to train the 1-bit CNN. There are two major challenges. One is how to compute the gradient of the sign function on activations, which is non-differentiable. The other is that the gradient of the loss with respect to the binary weight is too small to change the weight's sign. The authors of [7] proposed to adjust the standard SGD algorithm to approximately train the 1-bit CNN. Specifically, the gradient of the sign function on activations is approximated by the gradient of the piecewise linear function, as shown in Fig. 5(b). To tackle the second challenge, the method proposed in [7] updates the real-valued weights by the gradient computed with regard to the binary weight and obtains the binary weight by taking the sign of the real weights. As the identity shortcut will not add difficulty for training, the training algorithm proposed in [7] can also be adopted to train the Bi-Real net model. However, we propose a novel training algorithm to tackle the above two major challenges, which is more suitable for
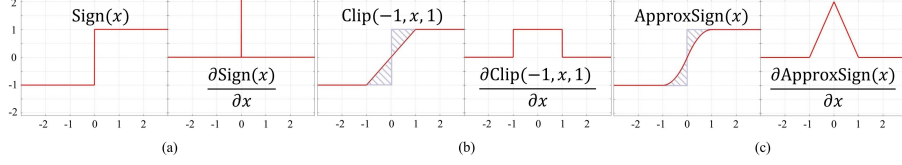
Fig. 5. (a) Sign function and its derivative, (b) Clip function and its derivative for approximating the derivative of the sign function, proposed in [7], (c) Proposed differentiable piecewise polynomial function and its triangle-shaped derivative for approximating the derivative of the sign function in gradients computation.

the Bi-Real net model as well as other 1-bit CNNs. Besides, we also propose a novel initialization method.

We present a graphical illustration of the training of Bi-Real net in Fig. 4. The identity shortcut is omitted in the graph for clarity, as it will not change the main part of the training algorithm.

Approximation to the derivative of the sign function with respect to activations. As shown in Fig. 5(a), the derivative of the sign function is an impulse function, which cannot be utilized in training.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial \mathbf{A}_b^{l,t}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial Sign(\mathbf{A}_r^{l,t})}{\partial \mathbf{A}_r^{l,t}} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l,t}} \frac{\partial F(\mathbf{A}_r^{l,t})}{\partial \mathbf{A}_r^{l,t}}, \qquad (2)$$

where $F(\mathbf{A}_r^{l,t})$ is a differentiable approximation of the non-differentiable $Sign(\mathbf{A}_r^{l,t})$. In [7], $F(\mathbf{A}_r^{l,t})$ is set as the clip function, leading to the derivative as a step-function (see 5(b)). In this work, we utilize a piecewise polynomial function (see 5(c)) as the approximation function, as Eq. (3) left.

$$F(a_r) = \begin{cases} -1 & \text{if } a_r < -1 \\ 2a_r + a_r^2 & \text{if } -1 \leqslant a_r < 0 \\ 2a_r - a_r^2 & \text{if } 0 \leqslant a_r < 1 \\ 1 & \text{otherwise} \end{cases}, \quad \frac{\partial F(a_r)}{\partial a_r} = \begin{cases} 2 + 2a_r & \text{if } -1 \leqslant a_r < 0 \\ 2 - 2a_r & \text{if } 0 \leqslant a_r < 1 \\ 0 & \text{otherwise} \end{cases},$$

$$(3)$$

As shown Fig. 5, the shaded areas with blue slashes can reflect the difference between the sign function and its approximation. The shaded area corresponding to the clip function is 1, while that corresponding to Eq. (3) left is $\frac{2}{3}$. We conclude that Eq. (3) left is a closer approximation to the sign function than the clip function. Consequently, the derivative of Eq. (3) left is formulated as Eq. (3) right, which is a piecewise linear function.

Magnitude-aware gradient with respect to weights. Here we present how to update the binary weight parameter in the lth block, *i.e.*, $\mathbf{W}_b^l \in \{-1, +1\}$. For clarity, we assume that there is only one weight kernel, *i.e.*, $\mathbf{W}_b^l$ is a matrix.

The standard gradient descent algorithm cannot be directly applied as the gradient is not large enough to change the binary weights. To tackle this problem, the method of [7] introduced a real weight $\mathbf{W}_r^l$ and a sign function during training. Hence the binary weight parameter can be seen as the output to the sign function, i.e., $\mathbf{W}_b^l = Sign(\mathbf{W}_r^l)$, as shown in the upper sub-figure in Fig. 4. Consequently, $\mathbf{W}_r^l$ is updated using gradient descent in the backward pass, as follows

$$\mathbf{W}_r^{l,t+1} = \mathbf{W}_r^{l,t} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_r^{l,t}} = \mathbf{W}_r^{l,t} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_b^{l,t}} \frac{\partial \mathbf{W}_b^{l,t}}{\partial \mathbf{W}_r^{l,t}}. \tag{4}$$

Note that $\frac{\partial \mathbf{W}_b^{l,t}}{\partial \mathbf{W}_r^{l,t}}$ indicates the element-wise derivative. In [7], $\frac{\partial \mathbf{W}_b^{l,t}(i,j)}{\partial \mathbf{W}_r^{l,t}(i,j)}$ is set to 1 if $\mathbf{W}_r^{l,t}(i,j) \in [-1,1]$, otherwise 0. The derivative $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_b^{l,t}}$ is derived from the chain rule, as follows

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_b^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l+1,t}} \frac{\partial \mathbf{A}_r^{l+1,t}}{\partial \mathbf{A}_m^{l,t}} \frac{\partial \mathbf{A}_m^{l,t}}{\partial \mathbf{W}_b^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l+1,t}} \theta^{l,t} \mathbf{A}_b^l, \tag{5}$$

where $\theta^{l,t} = \frac{\partial \mathbf{A}_r^{l+1,t}}{\partial \mathbf{A}_m^{l,t}}$ denotes the derivative of the BatchNorm layer (see Fig. 4) and has a negative correlation to $\mathbf{W}_b^{l,t}$. As $\mathbf{W}_b^{l,t} \in \{-1,+1\}$, the gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_r^{l,t}}$ is only related to the sign of $\mathbf{W}_r^{l,t}$, while is independent of its magnitude.

Based on this observation, we propose to replace the above sign function by a magnitude-aware function, as follows:

$$\overline{\mathbf{W}}_b^{l,t} = \frac{\parallel \mathbf{W}_r^{l,t} \parallel_{1,1}}{|\mathbf{W}_r^{l,t}|} Sign(\mathbf{W}_r^{l,t}), \tag{6}$$

where $|\mathbf{W}_r^{l,t}|$ denotes the number of entries in $\mathbf{W}_r^{l,t}$. Consequently, the update of $\mathbf{W}_r^l$ becomes

$$\mathbf{W}_r^{l,t+1} = \mathbf{W}_r^{l,t} - \eta \frac{\partial \mathcal{L}}{\partial \overline{\mathbf{W}}_b^{l,t}} \frac{\partial \overline{\mathbf{W}}_b^{l,t}}{\partial \mathbf{W}_r^{l,t}} = \mathbf{W}_r^{l,t} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l+1,t}} \overline{\theta}^{l,t} \mathbf{A}_b^l \frac{\partial \overline{\mathbf{W}}_b^{l,t}}{\partial \mathbf{W}_r^{l,t}}, \tag{7}$$

where $\frac{\partial \overline{\mathbf{W}}_b^{l,t}}{\partial \mathbf{W}_r^{l,t}} \approx \frac{\|\mathbf{W}_r^{l,t}\|_{1,1}}{|\mathbf{W}_r^{l,t}|} \cdot \frac{\partial Sign(\mathbf{W}_r^{l,t})}{\partial \mathbf{W}_r^{l,t}} \approx \frac{\|\mathbf{W}_r^{l,t}\|_{1,1}}{|\mathbf{W}_r^{l,t}|} \cdot \mathbf{1}_{|\mathbf{W}_r^{l,t}|<1}$ and $\overline{\theta}^{l,t}$ is associated with the magnitude of $\mathbf{W}_r^{l,t}$. Thus, the gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_r^{l,t}}$ is related to both the sign and magnitude of $\mathbf{W}_r^{l,t}$. After training for convergence, we still use $Sign(\mathbf{W}_r^l)$ to obtain the binary weight $\mathbf{W}_b^l$ (i.e., -1 or +1), and use θ^l to absorb $\frac{\|\mathbf{W}_r^l\|_{1,1}}{|\mathbf{W}_r^l|}$ and to associate with the magnitude of $\mathbf{W}_b^l$ used for inference.

Initialization. In [14], the initial weights of the 1-bit CNNs are derived from the corresponding real-valued CNN model pre-trained on ImageNet. However, the activation of ReLU is non-negative, while that of Sign is -1 or $+1$. Due

to this difference, the real CNNs with ReLU may not provide a suitable initial point for training the 1-bit CNNs. Instead, we propose to replace ReLU with $\text{clip}(-1, x, 1)$ to pre-train the real-valued CNN model, as the activation of the clip function is closer to the sign function than ReLU. The efficacy of this new initialization will be evaluated in experiments.

4   Experiments

In this section, we firstly introduce the dataset for experiments and implementation details in Sec 4.1. Then we conduct ablation study in Sec. 4.2 to investigate the effectiveness of the proposed techniques. This part is followed by comparing our Bi-Real net with other state-of-the-art binary networks regarding accuracy in Sec 4.3. Sec. 4.4 reports memory usage and computation cost in comparison with other networks.

4.1   Dataset and Implementation Details

The experiments are carried out on the ILSVRC12 ImageNet classification dataset [22]. ImageNet is a large-scale dataset with 1000 classes and 1.2 million training images and 50k validation images. Compared to other datasets like CIFAR-10 [11] or MNIST [18], ImageNet is more challenging due to its large scale and great diversity. The study on this dataset will validate the superiority of the proposed Bi-Real network structure and the effectiveness of three training methods for 1-bit CNNs. In our comparison, we report both the top-1 and top-5 accuracies.

For each image in the ImageNet dataset, the smaller dimension of the image is rescaled to 256 while keeping the aspect ratio intact. For *training*, a random crop of size 224 × 224 is selected. Note that, in contrast to XNOR-Net and the full-precision ResNet, we do not use the operation of random resize, which might improve the performance further. For *inference*, we employ the 224 × 224 center crop from images.

Training: We train two instances of the Bi-Real net, including an *18-layer Bi-Real net* and a *34-layer Bi-Real net*. The training of them consists of two steps: training the 1-bit convolution layer and retraining the BatchNorm. In the first step, the weights in the 1-bit convolution layer are binarized to the sign of real-valued weights multiplying the absolute mean of each kernel. We use the SGD solver with the momentum of 0.9 and set the weight-decay to 0, which means we no longer encourage the weights to be close to 0. For the 18-layer Bi-Real net, we run the training algorithm for 20 epochs with a batch size of 128. The learning rate starts from 0.01 and is decayed twice by multiplying 0.1 at the $10th$ and the $15th$ epoch. For the 34-layer Bi-Real net, the training process includes 40 epochs and the batch size is set to 1024. The learning rate starts from 0.08 and is multiplied by 0.1 at the $20th$ and the $30th$ epoch, respectively. In the second step, we constraint the weights to -1 and 1, and set the learning rate in all convolution layers to 0 and retrain the BatchNorm layer for 1 epoch to absorb the scaling factor.
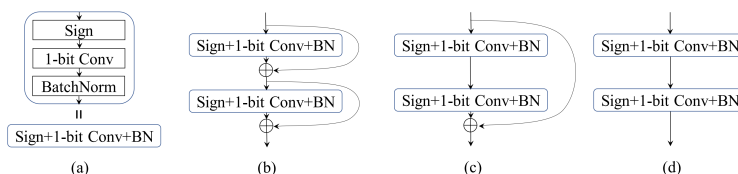
Fig. 6. Three different networks differ in the shortcut design of connecting the blocks shown in (a) conjoint layers of Sign, 1-bit Convolution, and the BatchNorm. (b) Bi-Real net with shortcut bypassing every block (c) Res-Net with shortcut bypassing two blocks, which corresponds to the ReLU-only pre-activation proposed in [6] and (d) Plain-Net without the shortcut. These three structures shown in (b), (c) and (d) have the same number of weights.

Inference: we use the trained model with binary weights and binary activations in the 1-bit convolution layers for inference.

4.2   Ablation Study

Three building blocks. The shortcut in our Bi-Real net transfers real-valued representation without additional memory cost, which plays an important role in improving its capability. To verify its importance, we implemented a Plain-Net structure without shortcut as shown in Fig. 6 (d) for comparison. At the same time, as our network structure employs the same number of weight filters and layers as the standard ResNet, we also make a comparison with the standard ResNet shown in Fig. 6 (c). For a fair comparison, we adopt the ReLU-only pre-activation ResNet structure in [6], which differs from Bi-Real net only in the structure of two layers per block instead of one layer per block. The layer order and shortcut design in Fig. 6 (c) are also applicable for 1-bit CNN. The comparison can justify the benefit of implementing our Bi-Real net by specifically replacing the 2-conv-layer-per-block Res-Net structure with two 1-conv-layer-per-block Bi-Real structure.

As discussed in Sec. 3, we proposed to overcome the optimization challenges induced by discrete weights and activations by 1) approximation to the derivative of the sign function with respect to activations, 2) magnitude-aware gradient with respect to weights and 3) clip initialization. To study how these proposals benefit the 1-bit CNNs individually and collectively, we train the 18-layer structure and the 34-layer structure with a combination of these techniques on the ImageNet dataset. Thus we derive $2 \times 3 \times 2 \times 2 \times 2 = 48$ pairs of values of top-1 and top-5 accuracy, which are presented in Table 1.

Based on Table 1, we can evaluate each technique's individual contribution and collective contribution of each unique combination of these techniques towards the final accuracy.

1) Comparing the $4^{th} - 7^{th}$ columns with the $8^{th} - 9^{th}$ columns, both the proposed Bi-Real net and the binarized standard ResNet outperform their plain

Table 1. Top-1 and top-5 accuracies (in percentage) of different combinations of the three proposed techniques on three different network structures, Bi-Real net, ResNet and Plain Net, shown in Fig.6.

Initiali-zation	Weight update	Activation backward	Bi-Real-18 top-1	top-5	Res-18 top-1	top-5	Plain-18 top-1	top-5	Bi-Real-34 top-1	top-5	Res-34 top-1	top-5	Plain-34 top-1	top-5
ReLU	Original	Original	32.9	56.7	27.8	50.5	3.3	9.5	53.1	76.9	27.5	49.9	1.4	4.8
		Proposed	36.8	60.8	32.2	56.0	4.7	13.7	58.0	81.0	33.9	57.9	1.6	5.3
	Proposed	Original	40.5	65.1	33.9	58.1	4.3	12.2	59.9	82.0	33.6	57.9	1.8	6.1
		Proposed	47.5	71.9	41.6	66.4	8.5	21.5	61.4	83.3	47.5	72.0	2.1	6.8
	Real-valued Net		68.5	88.3	67.8	87.8	67.5	87.5	70.4	89.3	69.1	88.3	66.8	86.8
Clip	Original	Original	37.4	62.4	32.8	56.7	3.2	9.4	55.9	79.1	35.0	59.2	2.2	6.9
		Proposed	38.1	62.7	34.3	58.4	4.9	14.3	58.1	81.0	38.2	62.6	2.3	7.5
	Proposed	Original	53.6	77.5	42.4	67.3	6.7	17.1	60.8	82.9	43.9	68.7	2.5	7.9
		Proposed	**56.4**	**79.5**	45.7	70.3	12.1	27.7	**62.2**	**83.9**	49.0	73.6	2.6	8.3
	Real-valued Net		68.0	88.1	67.5	87.6	64.2	85.3	69.7	89.1	67.9	87.8	57.1	79.9
Full-precision original ResNet[5]					69.3	89.2					73.3	91.3		

counterparts with a significant margin, which validates the effectiveness of short-cut and the disadvantage of directly concatenating the 1-bit convolution layers. As Plain-18 has a thin and deep structure, which has the same weight filters but no shortcut, binarizing it results in very limited network representational capacity in the last convolution layer, and thus can hardly achieve good accuracy.

2) Comparing the $4^{th} - 5^{th}$ and $6^{th} - 7^{th}$ columns, the 18-layer Bi-Real net structure improves the accuracy of the binarized standard ResNet-18 by about 18%. This validates the conjecture that the Bi-Real net structure with more shortcuts further enhances the network capacity compared to the standard ResNet structure. Replacing the 2-conv-layer-per-block structure employed in Res-Net with two 1-conv-layer-per-block structure, adopted by Bi-Real net, could even benefit a real-valued network.

3) All proposed techniques for initialization, weight update and activation backward improve the accuracy at various degrees. For the 18-layer Bi-Real net structure, the improvement from the weight (about 23%, by comparing the 2nd and 4th rows) is greater than the improvement from the activation (about 12%, by comparing the 2nd and 4th rows) and the improvement from replacing ReLU with Clip for initialization (about 13%, by comparing the 2nd and 7th rows). These three proposed training mechanisms are independent and can function collaboratively towards enhancing the final accuracy.

4) The proposed training methods can improve the final accuracy for all three networks in comparison with the original training method, which implies these proposed three training methods are universally suitable for various networks.

5) The two implemented Bi-Real nets (*i.e.* the 18-layer and 34-layer structures) together with the proposed training methods, achieve approximately 83% and 89% of the accuracy level of their corresponding full-precision networks, but with a huge amount of speedup and computation cost saving.

Table 2. This table compares both the top-1 and top-5 accuracies of our Bi-real net with other state-of-the-art binarization methods: BinaryNet [7] , XNOR-Net [19], ABC-Net [14] on both the Res-18 and Res-34 [5]. The Bi-Real net outperforms other methods by a considerable margin.

		Bi-Real net	BinaryNet	ABC-Net	XNOR-Net	Full-precision
18-layer	Top-1	56.4%	42.2%	42.7%	51.2%	69.3%
	Top-5	79.5%	67.1%	67.6%	73.2%	89.2%
34-layer	Top-1	62.2%	–	52.4%	–	73.3%
	Top-5	83.9%	–	76.5%	–	91.3%

In short, the shortcut enhances the network representational capability, and the proposed training methods help the network to approach the accuracy upper bound.

4.3   Accuracy Comparison With State-of-The-Art

While the ablation study demonstrates the effectiveness of our 1-layer-per-block structure and the proposed techniques for optimal training, it is also necessary to compare with other state-of-the-art methods to evaluate Bi-Real net's overall performance. To this end, we carry out a comparative study with three methods: BinaryNet [7], XNOR-Net [19] and ABC-Net [14]. These three networks are representative methods of binarizing both weights and activations for CNNs and achieve the state-of-the-art results. Note that, for a fair comparison, our Bi-Real net contains the same amount of weight filters as the corresponding ResNet that these methods attempt to binarize, differing only in the shortcut design.

Table 2 shows the results. The results of the three networks are quoted directly from the corresponding references, except that the result of BinaryNet is quoted from ABC-Net [14]. The comparison clearly indicates that the proposed Bi-Real net outperforms the three networks by a considerable margin in terms of both the top-1 and top-5 accuracies. Specifically, the 18-layer Bi-Real net outperforms its 18-layer counterparts BinaryNet and ABC-Net with relative 33% advantage, and achieves a roughly 10% relative improvement over the XNOR-Net. Similar improvements can be observed for 34-layer Bi-Real net. In short, our Bi-Real net is more competitive than the state-of-the-art binary networks.

4.4   Efficiency and Memory Usage Analysis

In this section, we analyze the saving of memory usage and speedup in computation of Bi-Real net by comparing with the XNOR-Net [19] and the full-precision network individually.

The memory usage is computed as the summation of 32 bit times the number of real-valued parameters and 1 bit times the number of binary parameters in the network. For efficiency comparison, we use FLOPs to measure the total

Table 3. Memory usage and FLOPs calculation in Bi-Real net.

		Memory usage	Memory saving	FLOPs	Speedup
18-layer	Bi-Real net	33.6 Mbit	11.14 $\times$	1.63 $\times 10^8$	11.06 $\times$
	XNOR-Net	33.7 Mbit	11.10 $\times$	1.67 $\times 10^8$	10.86 $\times$
	Full-precision Res-Net	374.1 Mbit	$-$	1.81 $\times 10^9$	$-$
34-layer	Bi-Real net	43.7 Mbit	15.97 $\times$	1.93 $\times 10^8$	18.99 $\times$
	XNOR-Net	43.9 Mbit	15.88 $\times$	1.98 $\times 10^8$	18.47 $\times$
	Full-precision Res-Net	697.3 Mbit	$-$	3.66 $\times 10^9$	$-$

real-valued multiplication computation in the Bi-Real net, following the calculation method in [5]. As the bitwise XNOR operation and bit-counting can be performed in a parallel of 64 by the current generation of CPUs, the FLOPs is calculated as the amount of real-valued floating point multiplication plus 1/64 of the amount of 1-bit multiplication.

We follow the suggestion in XNOR-Net [19], to keep the weights and activations in the first convolution and the last fully-connected layers to be real-valued. We also adopt the same real-valued 1x1 convolution in Type B short-cut [5] as implemented in XNOR-Net. Note that this 1x1 convolution is for the transition between two stages of ResNet and thus all information should be preserved. As the number of weights in those three kinds of layers accounts for only a very small proportion of the total number of weights, the limited memory saving for binarizing them does not justify the performance degradation caused by the information loss.

For both the 18-layer and the 34-layer networks, the proposed Bi-Real net reduces the memory usage by 11.1 times and 16.0 times individually, and achieves computation reduction of about 11.1 times and 19.0 times, in comparison with the full-precision network. Without using real-valued weights and activations for scaling binary ones during inference time, our Bi-Real net requires fewer FLOPs and uses less memory than XNOR-Net and is also much easier to implement.

5   Conclusion

In this work, we have proposed a novel 1-bit CNN model, dubbed Bi-Real net. Compared with the standard 1-bit CNNs, Bi-Real net utilizes a simple short-cut to significantly enhance the representational capability. Further, an advanced training algorithm is specifically designed for training 1-bit CNNs (including Bi-Real net), including a tighter approximation of the derivative of the sign function with respect the activation, the magnitude-aware gradient with respect to the weight, as well as a novel initialization. Extensive experimental results demonstrate that the proposed Bi-Real net and the novel training algorithm show superiority over the state-of-the-art methods. In future, we will explore other advanced integer programming algorithms (*e.g.*, Lp-Box ADMM [26]) to train Bi-Real Net.

References

1. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems. pp. 3123–3131 (2015)
2. Garg, R., BG, V.K., Carneiro, G., Reid, I.: Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: European Conference on Computer Vision. pp. 740–756. Springer (2016)
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
4. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
6. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
7. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 4107–4115. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf
8. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations (2016)
9. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
13. Lai, L., Suda, N., Chandra, V.: Deep convolutional neural network inference with floating-point weights and fixed-point activations. arXiv preprint arXiv:1703.03073 (2017)
14. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. In: Advances in Neural Information Processing Systems. pp. 345–353 (2017)
15. Liu, B., Wang, M., Foroosh, H., Tappen, M., Pensky, M.: Sparse convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 806–814 (2015)
16. Liu, F., Shen, C., Lin, G., Reid, I.D.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 2024–2039 (2016)
17. Luo, W., Sun, P., Zhong, F., Liu, W., Zhang, T., Wang, Y.: End-to-end active object tracking via reinforcement learning. ICML (2018)

18. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. vol. 2011, p. 5 (2011)
19. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision. pp. 525–542. Springer (2016)
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
21. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014)
22. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
24. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3476–3483 (2013)
25. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
26. Wu, B., Ghanem, B.: lp-box admm: A versatile framework for integer programming. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
27. Wu, B., Hu, B.G., Ji, Q.: A coupled hidden markov random field model for simultaneous face clustering and tracking in videos. Pattern Recognition **64**, 361–373 (2017)
28. Wu, B., Lyu, S., Hu, B.G., Ji, Q.: Simultaneous clustering and tracklet linking for multi-face tracking in videos. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2856–2863 (2013)
29. Zhang, H., Kyaw, Z., Chang, S.F., Chua, T.S.: Visual translation embedding network for visual relation detection. In: CVPR. vol. 1, p. 5 (2017)
30. Zhang, H., Kyaw, Z., Yu, J., Chang, S.F.: Ppr-fcn: Weakly supervised visual relation detection via parallel pairwise r-fcn. arXiv preprint arXiv:1708.01956 (2017)
31. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arXiv:1702.03044 (2017)
32. Zhou, E., Fan, H., Cao, Z., Jiang, Y., Yin, Q.: Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 386–391 (2013)
33. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
34. Zhu, X., Lei, Z., Liu, X., Shi, H., Li, S.Z.: Face alignment across large poses: A 3d solution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 146–155 (2016)