Learning Channel-wise Interactions for Binary Convolutional Neural Networks

Ziwei Wang[1,2,3], Jiwen Lu[1,2,3], Chenxin Tao[1], Jie Zhou[1,2,3], Qi Tian[4]
[1] Department of Automation, Tsinghua University, China
[2] State Key Lab of Intelligent Technologies and Systems, China
[3] Beijing National Research Center for Information Science and Technology, China
[4] Huawei Noah's Ark Lab, China
wang-zw18@mails.tsinghua.edu.cn; lujiwen@tsinghua.edu.cn; tcx16@mails.tsinghua.edu.cn;
jzhou@tsinghua.edu.cn; tian.qi@huawei.com

Abstract

In this paper, we propose a channel-wise interaction based binary convolutional neural network learning method (CI-BCNN) for efficient inference. Conventional methods apply xnor and bitcount operations in binary convolution with notable quantization error, which usually obtains inconsistent signs in binary feature maps compared with their full-precision counterpart and leads to significant information loss. In contrast, our CI-BCNN mines the channel-wise interactions through which prior knowledge is provided to alleviate inconsistency of signs in binary feature maps and preserves the information of input samples during inference. Specifically, we mine the channel-wise interactions by a reinforcement learning model, and impose channel-wise priors on the intermediate feature maps through the interacted bitcount function. Extensive experiments on the CIFAR-10 and ImageNet datasets show that our method outperforms the state-of-the-art binary convolutional neural networks with less computational and storage cost.

1. Introduction

Deep convolutional neural networks have achieved state-of-the-art performances in various vision applications such as object detection [10, 33, 22], tracking [13, 28, 1], face recognition [38, 29, 7] and many others. However, deploying deep convolutional neural networks in portable devices for inference is still limited because of the huge computational and storage cost. Moreover, high degrees of redundancy are exhibited in parameters of well-trained models [5]. Hence, it is desirable to design deep convolutional neural networks with fewer parameters and lighter architectures for efficient inference.

Recently, several neural network compression methods have been proposed including pruning [9, 21, 12], quantization [23, 17, 8], low-rank decomposition [6, 39, 43] and efficient architecture design [18, 15, 26]. Among these meth-

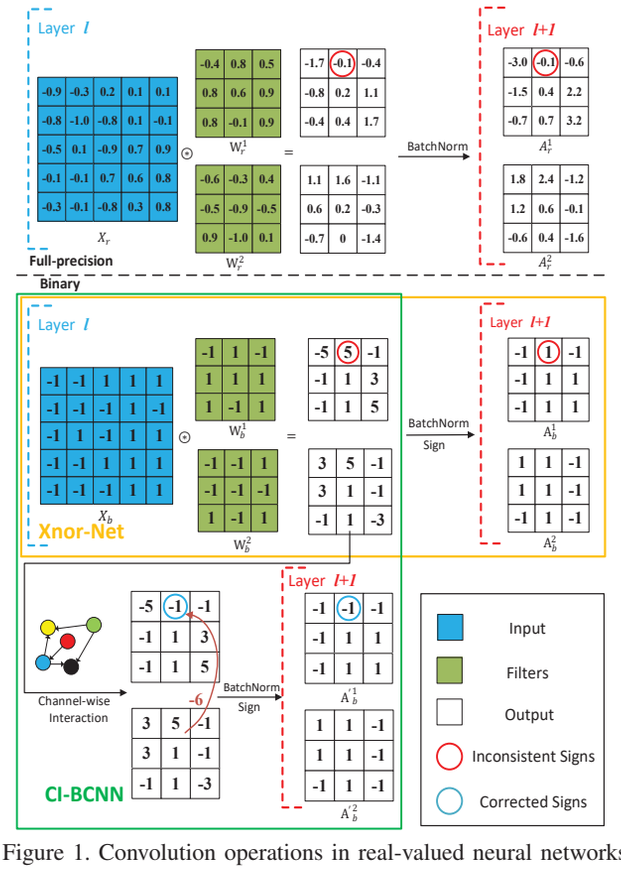


Figure 1. Convolution operations in real-valued neural networks (top), Xnor-Net (the yellow box) and our CI-BCNN (the green box). Because of the quantization error resulted from xnor and bitcount operations, Xnor-Net usually outputs binary feature maps which have the inconsistent signs compared with their full-precision counterpart (the red circle). Our CI-BCNN provides priors according to channel-wise interactions to correct the inconsistent signs (the blue circle), which preserves information for intermediate feature maps (best viewed in color).

ods, network quantization represents parameters of the neural networks in constrained bandwidth for faster processing and less memory consumption. Neural networks with binary weights replace multiply-accumulate operations with accumulation [3, 42, 14] to save storage cost and accelerate computation. However, real-valued calculation is still computationally expensive. To address this, neural networks with both binary weights and activations substitute multiply-accumulation with xnor and bitcount operations [32, 23, 24]. However, applying xnor and bitcount operations causes and accumulates notable quantization error, which usually results in inconsistent signs in binary feature maps compared with their full-precision counterpart. The information loss in binary neural networks explains the significant performance degradation compared with real-valued neural networks especially when evaluated in large-scale datasets such as ImageNet [4].

In this paper, we present a CI-BCNN method to learn binary neural network with channel-wise interactions for efficient inference. Unlike existing methods which directly apply xnor and bitcount operations, our method learns interacted bitcount according to the mined channel-wise interactions. The inconsistent signs in binary feature maps are corrected based on prior knowledge provided by channel-wise interactions, so that information of input images is preserved in the forward-propagation of binary neural networks. More specifically, we employ a reinforcement learning model to learn an directed acyclic graph for each convolutional layer, which stands for implicit channel-wise interactions. We obtain the interacted bitcount by adjusting the output of the original bitcount in line with the effects exerted by the graph. We train the binary convolutional neural network and the structure of graph simultaneously. Figure 1 depicts the comparison between our CI-BCNN and the conventional binary neural network, where inconsistent signs in binary feature maps are corrected according to the channel-wise interactions. Experiments on the CIFAR-10 [19] and ImageNet datasets show that our CI-BCNN outperforms most state-of-the-art binary neural networks by a large margin across various network architectures.

2. Related Work

Network Quantization: Network quantization has aroused extensive interest in machine learning and computer vision due to the reduction of the network complexity for wide deployment. Existing methods can be divided into two categories: neural networks with quantization on weights [3, 32, 42, 14] versus on both weights and activations [32, 17, 23, 24]. Weight-only quantization methods quantized weights in deep neural networks to save storage cost and substitute the original multiply-accumulation with accumulation for fast processing. Courbariaux *et al*. binarized the real-valued weights via a rigid sign function and obtained sufficiently high accuracy on small datasets. Rastegari *et al*. approximated the real-valued weights for binarization with a scaling factor to improve the accuracy. Zhang *et al*. trained an adaptive quantizer for weights according to their distribution, minimizing quantization error while staying compatible with the bitwise operations. Hou *et al*. applied the Taylor Expansion to minimize the loss caused by quantization perturbation, and proposed a proximal Newton algorithm to find the optimal solution for quantization strategy. Empirical studies showed that wider bandwidth for representing weights led to comparable performance with their full-precision counterpart, ternary and other multi-bit quantization methods [44, 36, 25] were proposed to obtain better performance. However, real-valued activations prevent substantial acceleration due to the existed accumulation operations. In the latter regard, weights and activations are both quantized so that multiply-accumulation is replaced by xnor and bitcount operations, leading to much less computational complexity. Rastegari *et al*. and Hubara *et al*. proposed neural networks with both weights and activations binarized, applying xnor and bitcount operations to substitute multiply-accumulation to obtain appreciable speedup. Lin *et al*. utilized more bases for weight and activation binarization, enhancing the performance especially in large-scale datasets. Liu *et al*. connected the real-valued activations of consecutive blocks with an identity shortcut before binarization to strengthen the representational capability of the network. They also used a new training algorithm to accurately back-propagate the gradient. Nevertheless, applying xnor and bitcount operations causes and accumulates the quantization error, leading to severe information loss because of inconsistent signs in binary feature maps compared with their real-valued counterpart.

Deep Reinforcement Learning: Deep reinforcement learning purposes to learn the policies for decision-making problems, which obtains promising results in playing games [27, 34], object detection [30, 31], visual tracking [16, 35, 40, 41] and many others. Recently, reinforcement learning has been adopted to network compression. Lin *et al*. applied a policy gradient model to judge the importance of feature maps, and pruned the network adaptively based on the input images and current feature maps to fully preserve the ability of the network. Ashok *et al*. shrank a large teacher network to a small student network by removing redundant layers and shrinking the size of the remaining layers, where a reinforcement learning model was employed to learn the policy. He *et al*. efficiently sampled the network architecture space by leveraging a reinforcement learning model, so that the model is compressed automatically without predefined pipelines. In this paper, we extend the reinforcement learning model to mine the channel-wise interactions in convolutional neural networks with bi-

nary weights and activations, through which the inconsistent signs caused by xnor and bitcount operations are corrected and information of input images is preserved in the forward-propagation process.

3. Approach

In this section, we first introduce neural networks with binary weights and activations briefly, which are efficient but suffer from inconsistent signs in intermediate feature maps. Then we present the details of imposing channel-wise interactions through the interacted bitcount. Finally, we propose a policy gradient model to mine the channel-wise interactions.

3.1. Binary Neural Networks

Let $\boldsymbol{W_r^l} \in \mathbb{R}^{w_f^l \times h_f^l}$ be the real-valued weights and $\boldsymbol{A_r^l} \in \mathbb{R}^{w_a^l \times h_a^l}$ be the full-precision activations of the l_{th} convolutional layer in a given L-layer CNN model, where (w_f^l, h_f^l) and (w_a^l, h_a^l) represents the width and height of filters and feature maps in the l_{th} layer. $\boldsymbol{A_r^l}$ carries information of input samples without binarization error:

$$\boldsymbol{A_r^l} = \boldsymbol{W_r^l} \circledast \boldsymbol{A_r^{l-1}}$$

where $\circledast$ stands for the standard convolution and activation layers are omitted for simplicity. In order to obtain neural networks with less computational and storage cost, we utilize binary weights and activations to replace the multiply-accumulation with xnor and bitcount operations [32] in the forward-propagation:

$$\boldsymbol{A_b^l} = sign(\boldsymbol{W_b^l} \odot \boldsymbol{A_b^{l-1}})$$

where $\boldsymbol{W_b^l} \in \{+1, -1\}^{w_f^l \times h_f^l}$ and $\boldsymbol{A_b^l} \in \{+1, -1\}^{w_a^l \times h_a^l}$ are binary weights and activations of the l_{th} layer respectively. $\odot$ indicates element-wise binary product representing xnor and bitcount operations in binary neural networks, where the bitcount is to count the number of ones in the results of xnor operations in each convolution. $sign$ means the sign function which maps number larger than one to one and to minus one otherwise.

The objective for binarizing convolutional neural networks is to minimize the distance between binary and real-valued feature maps so that information loss is minimal, which is written as follows:

$$\min_{\boldsymbol{W_b^l}, \boldsymbol{A_b^{l-1}}} ||\boldsymbol{A_r^l} - \boldsymbol{A_b^l}||_2^2 \tag{1}$$

where the optimization is NP-hard and the equivalent equation is $\boldsymbol{A_b^l} = sign(\boldsymbol{A_r^l})$. Conventional methods obtain the approximate solutions $\boldsymbol{W_b^l} = sign(\boldsymbol{W_r^l})$ and $\boldsymbol{A_b^{l-1}} = sign(\boldsymbol{A_r^{l-1}})$ by assuming:

$$\begin{aligned} sign(\boldsymbol{A_r^l}) &\approx sign(sign(\boldsymbol{W_r^l}) \circledast sign(\boldsymbol{A_r^{l-1}})) \\ &= sign(sign(\boldsymbol{W_r^l}) \odot sign(\boldsymbol{A_r^{l-1}})) \end{aligned}$$

However, due to the quantization error occurred in xnor and bitcount operations, the assumption does not always hold as shown in Figure 1. The approximate solution has inconsistent signs in $\boldsymbol{A_b^l}$ compared with $sign(\boldsymbol{A_r^l})$, so that Equation (1) is far from the optimal states. Moreover, the error is accumulated across layers and causes severe information loss of input images in the forward-propagation. Our objective is to minimize the difference between $\boldsymbol{A_b^l}$ and $sign(\boldsymbol{A_r^l})$ in each layer by correcting the inconsistent signs in $\boldsymbol{A_b^l}$.

3.2. Interacted Bitcount

Applying xnor operations brings significant quantization error compared with multiplication in full-precision. Moreover, original bitcount accumulate the error, which usually outputs inconsistent signs in feature maps compared to their real-valued counterpart. It is shown empirically that there is implicit dependency among filters, through which reliable priors are provided to offset the error resulted from xnor and bitcount operations. The interacted bitcount modifies the original bitcount as follows:

$$\widetilde{p}_{s,ij}^l = p_{s,ij}^l + \sum_t \delta_{ts}^l(p_{t,ij}^l) \tag{2}$$

where the upscript l represents the corresponding variable in the l_{th} convolutional layer. $p_{s,ij}^l$ and $p_{t,ij}^l$ are the integer pixel values output by the original bitcount in the i_{th} row and j_{th} column in the instructed (student) feature map F_s^l and directive (teacher) feature map F_t^l respectively. $\widetilde{p}_{s,ij}^l$ is the corresponding pixel value output by the interacted bitcount. δ_{ts}^l represents the influence function imposing on F_s^l from F_t^l.

To prevent the network suffering from heavy computation overhead of interacted bitcount, we simply design δ_{ts}^l as a discrete function. We partition the value range of pixels in F_t^l into $|K_{ts}^l|$ intervals with equal length when considering its interaction to F_s^l. K_{ts}^l is an odd integer so that no interaction exists if $p_{t,ij}^l$ stays near zero without sufficient information. Integer output of δ_{ts}^l is obtained as follows:

$$\begin{aligned} &\delta_{ts}^l(p_{t,ij}^l) = (\frac{1-|K_{ts}^l|}{2} + k) \cdot \frac{K_{ts}^l}{|K_{ts}^l|} \cdot [U_0 N_0], \\ &\text{if} \quad p_{t,ij}^l \in (p_k, p_{k+1}], \quad k = 0, 1..., |K_{ts}^l| - 1 \end{aligned} \tag{3}$$

where p_k is the origin of the k_{th} interval in value range partition of the teacher feature map F_t^l. N_0 is the maximum in value range of F_t^l, which is identical for all feature maps in the same layer. U_0 means the ratio of unit pixel modification to N_0, which is manually set to decide the importance of the prior. $[U_0 N_0]$ stands for the minimal integer larger than $U_0 N_0$. Meanwhile, K_{ts}^l can be a negative integer meaning that the student and teacher feature maps are negatively correlated. We have $|K_{ts}^l|$ choices from $\frac{1-|K_{ts}^l|}{2}[U_0 N_0]$ to $\frac{|K_{ts}^l|-1}{2}[U_0 N_0]$ for the output of δ_{ts}^l function, representing different effects on F_s^l exerted by F_t^l.

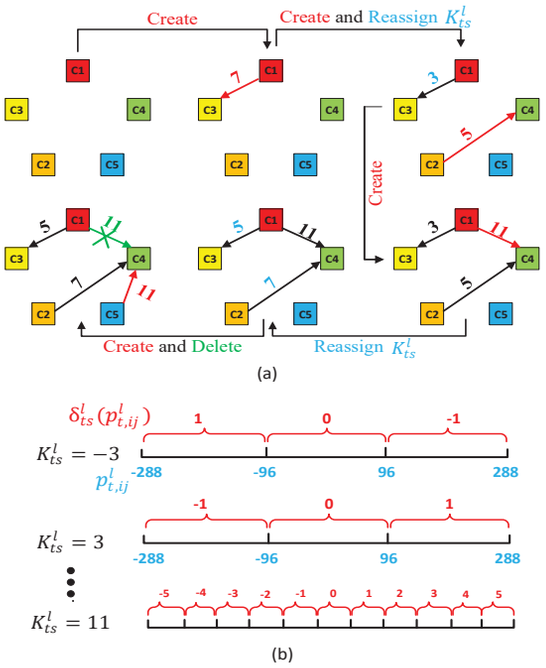


Figure 2. Illustrations for state transitions and interacted bitcount based on the mined graph. (a) An example for graph mining. We create edges, reassign K_{ts}^l and delete edges between different channels until finalizing the graph structure (best viewed in color). (b) A fast way to calculate the impact of graph on interacted bitcount according to pixel values in the teacher feature map via stair functions, where $N_0 = 288$ and U_0 is set as 0.001 in the example.

3.3. Channel-wise Interaction Mining via Policy Gradient

The channel-wise interaction is defined as edges in the graph among channels, which is expressed as existence and influence. Existence of an edge demonstrates the correlation between the two connected nodes, represented by one if the coherence is sufficiently significant and zero otherwise. An edge's influence means the impact on the end node imposed by the start node if the correlation exists. Because partitioning the value range of the teacher feature map into more intervals stands for greater impact of the channel-wise relationship, we depict the influence by K_{ts}^l. Mining the channel-wise interaction can be viewed as a Marcov Decision Process (MDP), defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}(\mathcal{S}, \mathcal{A}), \mathcal{R}(\mathcal{S}, \mathcal{A})\}$. At each step, the agent takes an action to create, delete or unchange edges to modify the existence of edges in the graph together with assigning different values to K_{ts}^l to represent various influences for all existing edges. The agent iteratively revamps the structure of the graph to maximize the gained reward until convergence or achieving the upper limit of steps.

States: The state space $\mathcal{S}$ expresses the current structure of the graph in all convolutional layers, which is represented as the direct product of the existence space $\mathcal{S}_e^l$ and influence space $\mathcal{S}_i^l$ across layer index l:

$$\mathcal{S} = \prod_{l=1}^{L} \mathcal{S}_e^l \times \mathcal{S}_i^l$$

where $\mathcal{S}_e^l$ is defined as the existence matrix $W_{es}^l \in \{1, 0\}^{c^l \times c^l}$ and c_l stands for the number of the channels in the l_{th} layer. For the element $w_{es,ts}^l$ in W_{es}^l, it equals to one if the directed interaction from t_{th} to s_{th} channel exists and equals to zero otherwise. Zero matrix in CI-BCNN is equivalent to conventional binary convolutional neural networks without channel-wise interaction. The influence space $\mathcal{S}_i^l$ is modeled by the influence matrix W_{is}^l with odd integers. In this paper, we limit the space with finite discrete numbers as $W_{is}^l \in \{\pm 3, \pm 5, \pm 7, ..., \pm(2K_0 + 1)\}^{c^l \times c^l}$, where K_0 is a hyperparameter representing the size of action space. In our implementation, element $w_{is,ts}^l$ in W_{is}^l is scaled to $\frac{K_{ts}^l}{|K_{ts}^l|} \cdot \frac{|K_{ts}^l| - 1}{2K_0}$ for regularization, which measures the impact of the corresponding interaction.

Action: The action set $\mathcal{A}$ is the direct product of action space in existence $\mathcal{A}_e^l$ and in influence $\mathcal{A}_i^l$ across all layers. $\mathcal{A}_e^l$ consists of three compositional sets: $\mathcal{A}_{e,c}^l$ for edge creation, $\mathcal{A}_{e,d}^l$ for edge deletion and $\{unchange\}$ for remaining the existence invariant. $\mathcal{A}_i^l$ depicts all possible odd integers in W_{is}^l for existing edges. Moreover, we stop the policy network when the graph converges or achieving the maximal steps. The whole action set is described as:

$$\begin{aligned}\mathcal{A} &= \prod_{i=1}^{L} (\mathcal{A}_e^l \times \mathcal{A}_i^l) \cup \{stop\} \\ &= \prod_{i=1}^{L} ((\mathcal{A}_{e,c}^l \cup \mathcal{A}_{e,d}^l \cup \{unchange\}) \times \mathcal{A}_i^l) \cup \{stop\}\end{aligned}$$

Figure 2 represents an example of stage transitions with actions and a fast way to implement interacted bitcount.

Transition Function: $\mathcal{T}(\mathcal{S}, \mathcal{A}) \rightarrow \mathcal{S}'$ is the transition function that shows the probability to convert the old state into the new one . $\mathcal{T}$ is constructed after defining state and action space, which is the direct product of two transition functions in all convolutional layers, $\mathcal{T}_e^l$ for existence transformation and $\mathcal{T}_i^l$ for influence change:

$$\mathcal{T}(\mathcal{S}, \mathcal{A}) = \prod_{i=1}^{L} \mathcal{T}_e^l(\mathcal{S}_e^l, \mathcal{A}_e^l) \times \mathcal{T}_i^l(\mathcal{S}_i^l, \mathcal{A}_i^l, \mathcal{S}_e^l, \mathcal{A}_e^l)$$

$\mathcal{T}_e^l$ is represented by a existence transition matrix $W_{et}^l \in [0, 1]^{c^l \times c^l}$, whose element $w_{et,ij}^l$ demonstrates the probability of connecting the directed edge from the i_{th} channel to the j_{th} one with the normalization $\sum_{i,j} w_{et,ij}^l = 1$. We select actions according to the following rules:

(1) *Create*: The density of the existence matrix ρ is defined as the ratio of ones in the existence matrix. When the

density of existence matrix is sparser than the hyper-parameter ρ_{max}, we create an edge directing to the j_{th} channel from the i_{th} one if the sampling strategy based on W^l_{et} selects the element $w^l_{et,ij}$ and the edge has not been connected.

(2) *Delete*: The probability of deletion is formulated as $W^{'l}_{et} = Norm([-\log w^l_{et,ij}]_{c_l \times c_l})$, where $Norm$ means the normalization operation that ensures $||W^{'l}_{et}||_1 = 1$. The probabilities of creation and deletion are negatively related because low probabilities of connection stand for the trend to disconnect the edges. Meanwhile, differences of low probabilities in W^l_{et} are very small and can only be revealed by their power exponent, so we applied logarithm to represent the possibility of deletion. We delete the existing edge between the i_{th} channel to the j_{th} one if the sampling strategy chooses the element $w^{'l}_{et,ij}$ in $W^{'l}_{et}$.

(3) *Unchange*: We remain the existence of edges unvaried if no creation or deletion happens.

As for the part of influence, we parameterize $\mathcal{T}^l_i$ with a influence matrix $W^l_{it} \in [-1, 1]^{c^l \times c^l}$ and select odd numbers deterministically in $\mathcal{A}^l_i$ for K_{ts} according to a stair function:

$$K^l_{ts} = \frac{w^l_{it,ts}}{|w^l_{it,ts}|} \cdot [2 * [|K_0 w^l_{it,ts}|] + 1] \tag{4}$$

Finally, we take the action *stop* to terminate the current epoch of channel-wise interaction mining when the policy network converges or achieves the maximal steps.

Reward Function: The reward function $\mathcal{R}(\mathcal{S}, \mathcal{A})$ in round τ is modeled as follows:

$$\begin{aligned} r(s_\tau, a_\tau) &= r_c(s_\tau, a_\tau) + r_p(s_\tau, a_\tau) \\ &= sgn(|C(s_\tau) - C(s_{\tau+1})| - h) \frac{C(s_\tau) - C(s_{\tau+1})}{|C(s_\tau) - C(s_{\tau+1})|} \\ &+ \frac{1}{N} \sum_{l=1}^{L} \sum_{t,s} \sum_{i,j} \frac{|p^l_{t,ij}(s_{\tau+1})| - |p^l_{s,ij}(s_{\tau+1})|}{||p^l_{t,ij}(s_{\tau+1})| - |p^l_{s,ij}(s_{\tau+1})||} \end{aligned}$$

where $C(s_\tau)$ represents the cross-entropy loss of the binary neural network for prediction under the graph mined in round τ, and h is a positive threshold whose value is assigned manually. $p^l_{t,ij}(s_{\tau+1})$ and $p^l_{s,ij}(s_{\tau+1})$ means the pixel values in the i_{th} row and j_{th} column of the student and teacher feature maps, which are output by the interacted bit-count with the graph mined in round $t+1$. N stands for the number of total pixels of feature maps in the binary neural networks, which equals to $\sum_{l=1}^{L} \sum_{t,s} \sum_{i,j} 1$.

The physical meaning of the reward function is illustrated by two terms. r_c encourages the graph imposed on the binary neural network to decrease the cross-entropy loss in classification. The agent acquires the reward $+1$ or -1 if the reduced or increased cross-entropy loss is larger than a set threshold h, while gains no reward when the loss does not change apparently. r_p aims to ensure that the teacher feature map is more informative than the student one so that reliable priors are provided. Because pixels carry more information are usually activated or deactivated significantly, we expect the mean absolute value in the teacher feature map is higher than their counterpart in the student one.

We employ a Encoder-decoder RNN for the policy network, which takes the current state of graph W^l_{es} and W^l_{is} as input, while outputs the transition matrix W^l_{et} and W^l_{it} for the binary convolutional layer. Figure 3 shows the overall framework for training our CI-BCNN with the policy network. We utilize the REINFORCE algorithm [37] to optimize the policy network. The objective is maximizing the expected return over the entire CI-BCNN learning process:

$$\max_\theta Z(\theta) = \mathbb{E}_\pi[\sum_{\tau=1}^{T} \gamma r_\tau(s_\tau, a_\tau)] \tag{5}$$

where θ means parameters in the policy network and π represents the selected policy. T stands for the time of sampling for each training batch and γ is the discount factor. According to the policy gradient method, we compute the expected gradient of the objective as follows:

$$\nabla_\theta Z = -\mathbb{E}_\pi[r_\tau(s_\tau, a_\tau) \nabla_\theta \log p(a_\tau | s_\tau)] \tag{6}$$

We apply Monte-Carlo Sampling to obtain the approximated gradients due to the intractability of exhaustion for all possible states. Meanwhile, $p(a_\tau|s_\tau)$ is entangled by actions for exploring edge existence and influence, and the probability to choose influence is deterministic and non-differentiable. In order to back-propagate gradients, we approximate the optimization problem as another differentiable one (formulated in supplementary materials).

4. Experiments

In this section, we evaluated our method on two datasets for image classification: CIFAR-10 and ImageNet. We firstly introduced the implementation details of our CI-BCNN and illustrated the effectiveness and intuitive logic of CI-BCNN by toy examples. Then we investigated the influence of hyperparameters by ablation study and compared the proposed CI-BCNN with state-of-the-art binarized neural networks regarding the accuracy. Finally, we analyzed the storage and computation complexity during inference in comparison with other methods.

4.1. Implementation Details

We trained our CI-BCNN with the VGG-small [42] and ResNet20 architectures on the CIFAR-10 dataset. We employed ResNet18 and ResNet34 for the proposed CI-BCNN in the experiments on the ImageNet dataset. We iteratively

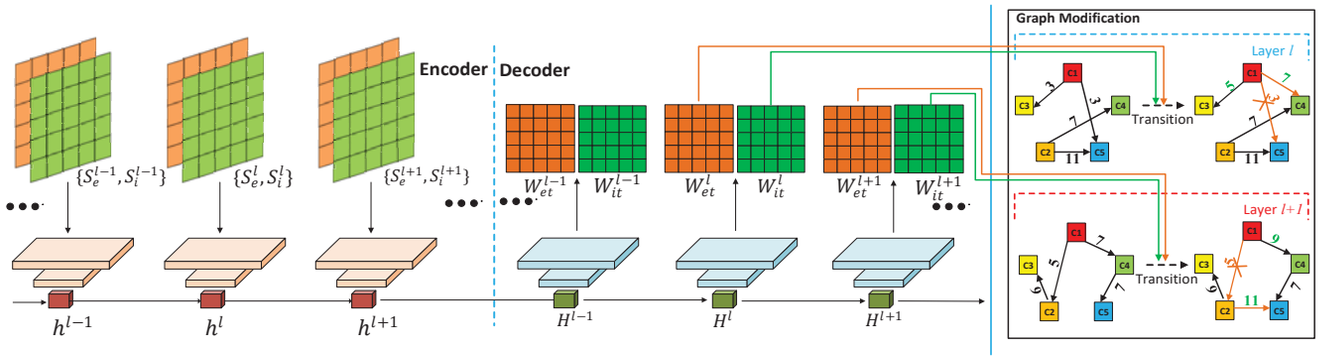


Figure 3. Overall framework for training the CI-BCNN. The left part is the policy network which consists of encoders and decoders. The encoders take the state of each layer as input and decoders output the corresponding transition matrix according to the hidden variables. The right part stands for the graphs in binary neural networks, where the l_{th} convolutional layer modifies its graph structure based on the transition matrix W_{et}^{l} and W_{it}^{l} (best viewed in color).

trained the binary neural network and the agent for mining channel-wise interactions in our CI-BCNN. In the training of the binary neural networks, the weights were binarized to the sign of real-valued weights multiplying the absolute mean value of each kernel. We followed the suggestion in XNOR-net [32] to keep the weights in the first and last layer real-valued. We used the Adam optimizer for all experiments with the batchsize 128. For experiments on CIFAR-10, we ran our algorithm for 100 epochs. The initial learning rate was set as 0.001 and decayed by multiplying 0.1 in the $50th$ and $90th$ epoch. In the training on ImageNet, We set the initial learning rate as 0.001 with multiplying 0.1 in the $20th$ and $30th$ epochs out of the total 40 epochs for ResNet18. The learning rate started from 0.005 with decay by $10\times$ in the $40th$ and $60th$ epochs out of the total 80 epochs for ResNet34. When finishing training, we froze all convolutional layers with the constrained weights to -1 and $+1$, and retrained the BatchNorm layer for 1 epoch to absorb the scaling factor.

When training the policy network, we applied two convolutional layers with a fully-connected layer for the encoder and used a fully-connected layer with two deconvolutional layers for the decoder in each module of RNN. We used $\frac{c_l}{16}$ matrices of the size 16×16 to represent state and transition matrices in the l_{th} layer for memory saving and computation acceleration. We set the hyperparameters U_0, ρ_{max}, K_0 and α as 0.01, 0.1, 2 and 0.001 respectively in the comparison with state-of-the-art methods.

4.2. Toy Examples

The thought of the proposed CI-BCNN is to mine the correlational graph among channels to correct inconsistent signs in binary feature maps caused by the xnor and bitcount operations. We conducted simple experiments a on the MNIST dataset [20] to show the correctness of our thoughts with intuition.

Effectiveness of the channel-wise interaction: We argue there are implicit correlations among channels, providing priors for eliminating significant quantization error which causes inconsistent signs of pixel values in intermediate feature maps. Through our policy gradient network, we can mine the relationships among channels. To validate our thoughts, we designed the architecture with two convolutional layers and one fully-connected layer for our CI-BCNN. Figure 4 shows the square of correlation coefficients among different channels in the binary convolutional layer, where darker colors mean higher correlations. The blue box represents the channel-wise interactions learned by our policy network. As can be seen, our model mined most significant correlations without irrelevant channels, which provides reliable priors for the interacted bitcount.

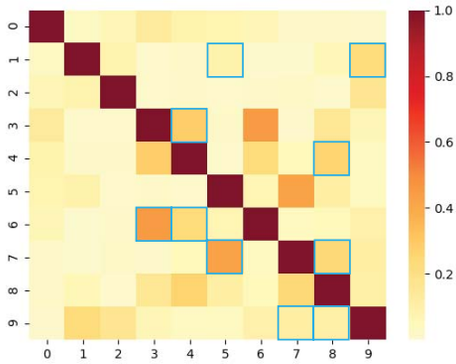


Figure 4. The square of correlation coefficients among 10 channels in the binary convolutional layer. Darker colors represent higher correlations and the blue box demonstrates the connections mined by our policy gradient networks.

Effectiveness of the interacted bitcount: Our interacted bitcount utilized channel-wise interaction to provide priors for recovering the original signs in the binary feature map. We expect that more pixels in binary feature maps have the same signs with their full-precision counterpart, so that information of input images is preserved during infer-

Table 1. Comparison on the ratio of pixels with consistent signs in different layers and corresponding accuracies of CI-BCNN.

	Conv2	Conv3	Acc.(%)
Bitcout	0.6238	0.6061	99.01
Interacted Bitcount	0.6638	0.6244	99.10

ence. Table 1 shows the effect of the interacted bitcount by the ratio of pixels with consistent signs in binary layers and the classification accuracy. The quantization error accumulates with the depth of layers, as the ratio is lower in the Conv3 layer compared with the Conv2 layer. Our CI-BCNN increases the ratio, which benefits from the priors provided by interacted bitcount.

4.3. Performance Analysis

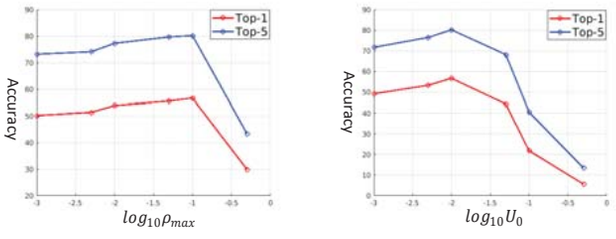
In order to investigate the effect of channel-wise interactions on intermediate feature maps, we conducted ablation study with varying maximal densities of the existence matrix ρ and ratios of unit pixel modification U_0. We reported the classification top-1 and top-5 accuracies on the ImageNet dataset with the ResNet18 architecture.

Performances w.r.t. the maximal density of existence matrix ρ: The density of existence matrix ρ is defined as the proportion of ones in the matrix, which is positively correlated with the hyperparameter ρ_{max}. Higher density of existence matrix represents more channel-wise interactions in the interacted bitcount. By changing the value of ρ_{max} in the training of the policy network, we can control the final density of existence matrix. The impact of ρ_{max} on the performance is illustrated in Figure 5(a). Medium density provides reliable priors for feature maps suffered from inconsistent signs. High density assigns excess connections in the graph with untrustworthy priors. Low density fails to consider priors, which is unable to alleviate inconsistent signs in binary feature maps caused by xnor and bitcount operations.

Performances w.r.t. the ratio of unit pixel modification U_0: Larger U_0 in the interacted bitcount stands for more significant modification, resulting in higher importance for the prior knowledge provided by channel-wise interactions. The prior knowledge becomes more important in classification when compared with the posterior information gained from input images. Figure 5(b) shows the performance versus different U_0. Medium U_0 provides prior knowledge for binarized neural networks, which is combined with posteriors learned from the input image. Large U_0 enforces too strong priors on feature maps, ignoring the knowledge obtained from the input sample. On the contrary, small U_0 fails to impose affective priors on intermediate feature maps which suffer from inconsistent signs.

4.4. Comparison with State-of-the-art Methods

In this section, we compared the performance of our CI-BCNN with existing methods including BNN [17], BC [3], BWN [32], Xnor-Net [32], Bi-Real-Net [24], ABC-Net [23], LQ-Nets [42], SYQ [8] HWGQ [2] and TTQ [44] through various architectures in image classification tasks on the CIFAR-10 and ImageNet datasets.



(a) Varying ρ_{max} (b) Varying U_0

Figure 5. Top-1 and top-5 classification accuracies on the ImageNet dataset of the CI-BCNN in the architecture of ResNet18 with (a) varying ρ_{max} and (b) varying U_0. Variables are represented by their logarithm.

Table 2. Comparison of classification accuracy (%) on CIFAR-10 with state-of-the-art methods in VGG-small and ResNet20.

Methods	Bitwidth (W/A)	VGG-small	ResNet20
Full-precision	32/32	93.20	92.10
BC	1/32	90.10	–
TTQ	2/32	–	91.13
HWGQ	1/2	92.50	–
LQ-Net	1/2	93.40	88.40
BNN	1/1	89.90	–
Xnor-Net	1/1	89.80	–
CI-BCNN	1/1	**92.47**	**91.10**

Comparison on CIFAR-10: The CIFAR-10 dataset consists of 60,000 images of size 32×32, which are divided into 10 categories. We applied 50,000 images as training set and the rest 10,000 as the test set. We padded 4 pixels on each side of the image and randomly cropped it into the size of 32×32. Meanwhile, we scaled and biased all images into the range $[-1, 1]$. We compared the accuracies of VGG-small [42] and ResNet20 quantized by different methods. Table 2 shows the results. The comparison clearly indicates the proposed CI-BCNN outperforms the existed neural networks with one-bit weight and activations by a sizable margin. Our method is even comparable with HWGQ which has activations in two bits and TTQ which has 2-bit weights and real-valued activations in VGG-small and ResNet20 architectures respectively.

Comparison on ImageNet: ImageNet (ILSVRC12) contains approximately 1.2 million training and 50K validation images from 1,000 categories. ImageNet is much more challenging because of its large scale and high diversity. Followed by data augmentation of bias subtraction applied in CIFAR-10, a 224×224 region is randomly cropped for training from the resized image whose shorter side is

Table 3. Comparison of classification accuracy (%) on ImageNet with state-of-the-art methods in ResNet18 and ResNet34.

Methods	Bitwidth (W/A)	ResNet18		ResNet34	
		top-1	top-5	top-1	top-5
Full-precision	32/32	69.30	89.20	73.30	91.30
BWN	1/32	60.80	83.00	60.80	83.00
TTQ	2/32	66.60	87.20	–	–
HWGQ	1/2	59.60	82.20	64.30	85.70
LQ-Net	1/2	62.60	84.30	66.60	86.90
SYQ	1/2	55.40	78.60	–	–
BNN	1/1	42.20	67.10	–	–
Xnor-Net	1/1	51.20	73.20	–	–
ABC-Net	1/1	42.70	67.60	52.40	76.50
Bi-Real-Net	1/1	56.40	79.50	62.20	83.90
CI-BCNN	1/1	**56.73**	**80.12**	**62.41**	**84.35**
CI-BCNN (add)	1/1	**59.90**	**84.18**	**64.93**	**86.61**

Table 4. Comparison of storage cost and FLOPs of different methods in ResNet18 and ResNet34.

		Storage Cost	FLOPs
ResNet18	Full-precision	374.1Mbit	1.81×10^9
	Xnor-Net	33.7Mbit	1.67×10^8
	Bi-Real-Net	33.6Mbit	1.63×10^8
	CI-BCNN	33.5Mbit	1.54×10^8
ResNet34	Full-precision	697.3Mbit	3.66×10^9
	Xnor-Net	43.9Mbit	1.98×10^8
	Bi-Real-Net	43.7Mbit	1.93×10^8
	CI-BCNN	43.5Mbit	1.82×10^8

256. For inference, we employed the 224×224 center crop from images. As demonstrated in [24], additional shortcut in every consecutive convolutional layers enhance the performance of binary neural networks, we employed extra shortcut for adjacent layers to further improve our CI-BCNN. We compared our CI-BCNN with state-of-the-art network quantization methods in ResNet18 and ResNet34 architectures and reported top-1 and top-5 accuracies in Table 3, where CI-BCNN (add) means our binary neural network with addtional shortcut applied in [24]. Bi-Real-Net achieves the outstanding performances among neural networks with binary weights and activations by adding more shortcuts and training with more accurate gradients. However, Bi-Real-Net fails to consider the quantization error caused by xnor and bitcount operations, which leads to inconsistent signs in binary feature maps with significant information loss. CI-BCNN preserves the information of input samples during inference through the interacted bitcount and the mined channel-wise interactions. Experiments on the ImageNet dataset shows the superiority of interacted bitcount directed by channel-wise interactions. Moreover, CI-BCNN obtains higher accuracies compared with HWGQ and BWN, which employs two-bit and float activations. In short, CI-BCNN is more competitive than the state-of-the-art neural networks with binary weights and activations.

4.5. Complexity Analysis

We analyzed the computational and storage complexity in comparison of Bi-Real Net, Xnor-Net and full-precision networks to show the saving of memory and speedup during inferences. The memory usage is represented by the storage for parameters of networks, which is calculated as summation of 32 bits time real-valued parameters and 1 bit times binary parameters. We use FLOPs to measure the computational complexity, following the calculation method in [11]. Because current generation of CPUs can implement 64 binary operations parallel in one block, the total FLOPs is calculated as the number of floating point multiplications plus $\frac{1}{64}$ of the amount of binary multiplications. The results are illustrated in Table 4 with our implementation settings.

The proposed CI-BCNN saves the storage cost by $11.17\times$ and $16.03\times$ in ResNet18 and ResNet34 respectively, and speeds up the computation by $11.75\times$ and $20.11\times$ in the above architectures when compared with the full-precision networks. In CI-BCNN, the storage overhead is negligible because the additional parameters are only the binary existence matrix and the discrete influence matrix stored in low bits. Meanwhile, the extra computation cost is resulted from interacted bitcount, which is insignificant compared with standard binary convolutions. On the contrary, our CI-BCNN saves computation and storage cost because scaling factors for weights and activations are removed compared with Xnor-Net, and the real-valued accumulation and batch normalization in extra shortcuts are not used in comparison with Bi-Real-Net. Generally speaking, CI-BCNN requires less memory usage and fewer FLOPs.

5. Conclusion

In this paper, we have proposed a binary convolutional neural network method called CI-BCNN for efficient inference. The proposed CI-BCNN mines the graph structure among channels via policy gradient and imposes channel-wise interactions by interacted bitcount, through which inconsistent signs in binary feature maps are corrected and information of input images is preserved during inference. Extensive experimental results demonstrate effectiveness of the proposed approach.

Acknowledgement

This work was supported in part in part by the National Key Research and Development Program of China under Grant 2016YFB1001001, in part by the National Natural Science Foundation of China under 61822603, Grant U1813218, Grant U1713214, Grant 61672306, Grant 61572271, and in part by the Shenzhen Fundamental Research Fund (Subject Arrangement) under Grant JCYJ20170412170602564.

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016.

[2] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, pages 5406–5414, 2017.

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[5] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *NIPS*, pages 2148–2156, 2013.

[6] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, pages 1269–1277, 2014.

[7] Changxing Ding and Dacheng Tao. Trunk-branch ensemble convolutional neural networks for video-based face recognition. *TPAMI*, 40(4):1002–1014, 2018.

[8] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *CVPR*, pages 4300–4309, 2018.

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2980–2988, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, volume 2, 2017.

[13] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, pages 597–606, 2015.

[14] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

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

[16] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, pages 105–114, 2017.

[17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

[18] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid nam2016learningnetworks for object detection. In *CVPR*, pages 936–944, 2017.

[23] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NIPS*, pages 344–352, 2017.

[24] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. *arXiv preprint arXiv:1808.00278*, 2018.

[25] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *NIPS*, pages 3290–3300, 2017.

[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 2018.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[28] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016.

[29] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *CVPR*, pages 3406–3415, 2017.

[30] Aleksis Pirinen and Cristian Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. In *CVPR*, pages 6945–6954, 2018.

[31] Yongming Rao, Dahua Lin, Jiwen Lu, and Jie Zhou. Learning globally optimized object detector via policy gradient. In *CVPR*, pages 6190–6198, 2018.

[32] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542, 2016.

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural

networks and tree search. *nature*, 529(7587):484, 2016.

[35] James Steven Supancic III and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *ICCV*, pages 322–331, 2017.

[36] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[37] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[38] Jiaolong Yang, Peiran Ren, Dongqing Zhang, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. Neural aggregation network for video face recognition. In *CVPR*, pages 4362–4371, 2017.

[39] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, pages 7370–7379, 2017.

[40] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, pages 1349–1358, 2017.

[41] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.

[42] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *arXiv preprint arXiv:1807.10029*, 2018.

[43] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *TPAMI*, 38(10):1943–1955, 2016.

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