Learning Channel-Wise Interactions for Binary Convolutional Neural Networks

Ziwei Wang, *Student Member, IEEE*, Jiwen Lu, *Senior Member, IEEE*, and Jie Zhou, *Senior Member, IEEE*

Abstract—In this paper, we propose a channel-wise interaction based binary convolutional neural networks (CI-BCNN) approach for efficient inference. Conventional binary convolutional neural networks usually apply the xnor and bitcount operations in the binary convolution with notable quantization errors, which obtain opposite signs of pixels in binary feature maps compared to their full-precision counterparts and lead to significant information loss. In our proposed CI-BCNN method, we exploit the channel-wise interactions with the prior knowledge which aims 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 using a reinforcement learning model, and impose channel-wise priors on the intermediate feature maps to correct inconsistent signs through the interacted bitcount. Since CI-BCNN mines the channel-wise interactions in a large search space where each channel may correlate with others, the search deficiency caused by sparse interactions obstacles the agent to obtain the optimal policy. To address this, we further present a hierarchical channel-wise interaction based binary convolutional neural networks (HCI-BCNN) method to shrink the search space via hierarchical reinforcement learning. Moreover, we propose a denoising interacted bitcount operation in binary convolution by smoothing the channel-wise interactions, so that noise in channel-wise priors can be alleviated. Extensive experimental results on the CIFAR-10 and ImageNet datasets demonstrate the effectiveness of the proposed CI-BCNN and HCI-BCNN.

Index Terms—Binary convolutional neural networks, channel-wise interactions, deep reinforcement learning, hierarchical reinforcement learning, feature denoising

1 Introduction

Deep convolutional neural networks have achieved state-of-the-art performance in various vision applications such as object detection [18], [35], [51], tracking [4], [21], [45], face recognition [11], [46], [64] 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 [9]. 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 [16], [19], [33], quantization [13], [28], [36], low-rank decomposition [10], [65], [69], distillation [1], [2], [42] and efficient architecture design [23], [29], [41]. Among these methods, network quantization represents parameters of neural networks in constrained bandwidth for faster processing and less memory consumption. Neural networks with binary weights estimate multiply-accumulate operations with addition and subtraction [7], [22], [68]. 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 [36], [38], [50]. As proven in [50], the information loss in binary convolutional neural networks is minimized when elements in binary feature maps have the consistent signs with their real-valued counterparts. However, applying xnor and bitcount operations usually causes and accumulates notable quantization error, which results in inconsistent signs between binary and full-precision feature maps. The information loss from binarized neural networks causes the significant performance degradation especially in large-scale datasets such as ImageNet [8].

Since interdependencies among channels have been shown in neural networks [24], [47], the feature response in different channels is correlated. The pixel value of one feature map can provide priors for that in its correlated feature map, which is defined as the channel-wise interaction. In this paper, we present a channel-wise interaction based convolutional neural networks method with binary weights and activations for efficient inference. Unlike existing methods which directly apply xnor and bitcount operations, our method learns the channel-wise interactions to correct the inconsistent signs in binary feature maps and decrease information loss. Fig. 1 shows the conventional and the channel-wise

- *Z. Wang and J. Lu are with the State Key Lab of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: wang-zw18@mails.tsinghua.edu.cn, lujiwen@tsinghua.edu.cn.*
- *J. Zhou is with the State Key Lab of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology (BNRist), the Department of Automation, Tsinghua University, Beijing 100084, China, and also with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China. E-mail: jzhou@tsinghua.edu.cn.*

Manuscript received 31 July 2019; revised 23 Mar. 2020; accepted 12 Apr. 2020. Date of publication 20 Apr. 2020; date of current version 2 Sept. 2021. (Corresponding author: Jiwen Lu.) Recommended for acceptance by B. Ommer. Digital Object Identifier no. 10.1109/TPAMI.2020.2988262

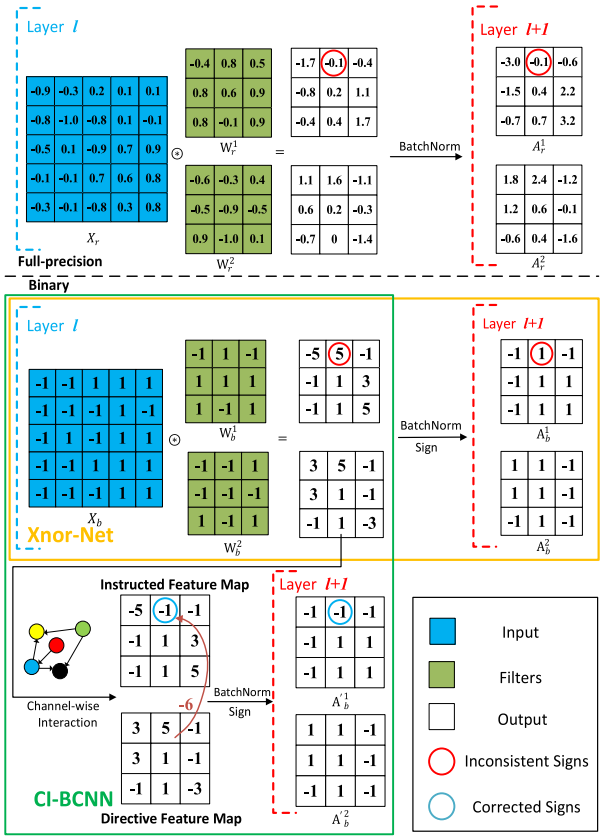


Fig. 1. Convolution operations in real-valued neural networks (top), Xnor-Net (the yellow box) and our CI-BCNN (the green box). Because of the quantization error resulted from xnor and bitcount operations, Xnor-Net usually outputs binary feature maps which have inconsistent signs compared with their full-precision counterparts (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).

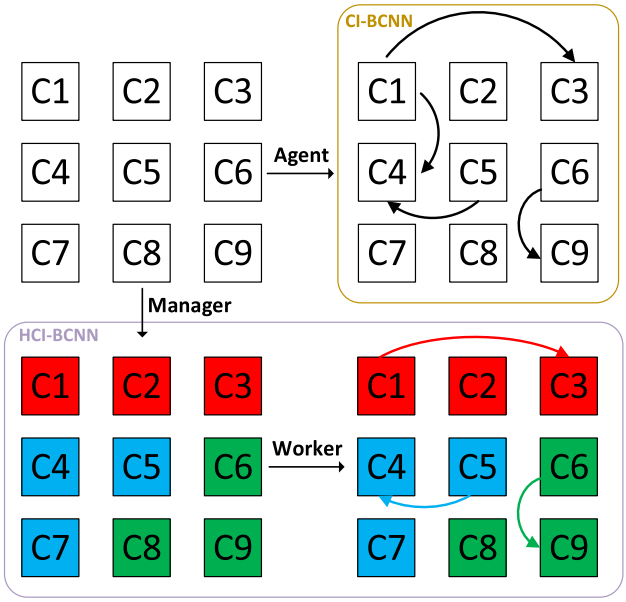


Fig. 2. Differences between the proposed CI-BCNN and HCI-BCNN, where $C_1 \sim C_9$ represent different channels in a convolutional layer. CI-BCNN mines the channel-wise interactions in large space where each channel may be correlated with others, and the search deficiency obstacles the agent to learn the optimal policy. On the contrary, HCI-BCNN obtains the channel-wise priors hierarchically in a shrunk policy space and improves search efficiency, where the manager partitions channels into different correlated groups (marked by various colors) and the worker searches for fine-grained channel-wise interactions for each group (best viewed in color).

interaction based binary convolutional neural networks. As exhaustively searching for the optimal channel-wise interactions is NP-hard, we employ the reinforcement learning method to solve the problem with acceptable cost. More specifically, we employ policy gradient to learn a directed acyclic graph for each convolutional layer, which stands for implicit channel-wise interactions. The mined channel-wise interactions correct the inconsistent signs in binary feature maps through the proposed interacted bitcount (IB), which adds a penalty to the conventional bitcount by considering the feature response in correlated channels. We train the binary convolutional neural networks and the structure of graph iteratively.

In fact, the channel-wise interactions are sparse because the independencies among channels are required to maintain the representational capability of binary convolutional neural networks. While CI-BCNN mines the channel-wise interactions in a large search space where each channel may correlate with others, the search deficiency caused by sparse correlation obstacles us to obtain the optimal channel-wise interactions. In order to address these limitations, we further propose a hierarchical channel-wise interaction based binary convolutional neural networks (HCI-BCNN) to shrink the search space and improve the search efficiency. More specifically, we employ a hierarchical reinforcement learning (HRL) model to mine the channel-wise interactions, where the agent in the top level (manager) partitions correlated channels and the low-level agent (worker) learns the fine-grained channel-wise interactions for each group. Fig. 2 demonstrates the difference between CI-BCNN and HCI-BCNN. Meanwhile, noise in channel-wise interactions corrupts the information in intermediate feature maps and degrades the performance. We present denoising interacted bitcount (DIB) to smooth the channel-wise interactions with feature denoising techniques [61], [63], so that robust channel-wise priors contribute to alleviating inconsistent signs. Extensive experiments on the CIFAR-10 [30] and ImageNet datasets show that our CI-BCNN and HCI-BCNN outperform most state-of-the-art binary neural networks across various network architectures.

This paper is an extended version of our conference paper [60], where we make the following new contributions in this journal paper:

1) We propose a new HCI-BCNN method based on CI-BCNN which was proposed in the conference version by mining the channel-wise interactions hierarchically with HRL, so that the search space shrinks significantly and the optimal channel-wise interactions are effectively selected.
2) We present denoising interacted bitcount to smooth the channel-wise interactions with feature denoising operations, so that channel-wise priors are more robust to noise.
3) We conduct extensive experiments on CIFAR-10 and ImageNet to evaluate the proposed CI-BCNN and

HCI-BCNN, and the results show the effectiveness and the efficiency of the presented methods.

2 Related Work

In this section, we briefly review four related topics: 1) network quantization, 2) deep reinforcement learning, 3) hierarchical reinforcement learning and 4) feature denoising.

2.1 Network Quantization

Network quantization has aroused extensive interest in machine learning and computer vision due to the reduction of the network complexity. Existing methods can be divided into two categories: neural networks with quantization on weights [7], [50], [22], [68] versus on both weights and activations [28], [36], [38], [50]. Weight-only quantization methods quantize weights in deep neural networks to save storage cost and evaluate the original multiply-accumulation by accumulation for fast processing. Courbariaux *et al.* [7] binarized the real-valued weights via the sign function and obtained high accuracy on small datasets. Zhang *et al.* [68] trained an adaptive quantizer for weights according to their distribution, minimizing quantization error while staying compatible with the bitwise operations. Hou *et al.* [22] 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. Since empirical studies showed that wider bandwidth for representing weights led to comparable performance with their full-precision counterparts, ternary and other multi-bit quantization methods [39], [56], [70] 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.* [50] and Hubara *et al.* [28] proposed neural networks with both weights and activations binarized, applying xnor and bitcount operations to substitute multiply-accumulation for speedup. Lin *et al.* [36] utilized more bases for weight and activation binarization, enhancing the performance especially in large-scale datasets. Liu *et al.* [37] applied the circulant filters, convolution and back propagation to enhance the capacity of the networks. Liu *et al.* [38] 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. Furthermore, Bethge *et al.* [5] conducted empirical study to show that maintaining rich information flow in binary neural networks is important, so that they proposed BinaryDenseNet with real-valued down-sampling layers. 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 counterparts.

2.2 Deep Reinforcement Learning

Deep reinforcement learning purposes to learn the policies for decision-making problems with deep neural networks, which obtains promising results in playing games [43], [52], object detection [48], [49], visual tracking [26], [53], [66], [67], network architecture search [3], [71] and many others. Recently, deep reinforcement learning has been adopted to network compression. Lin *et al.* [34] 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 with fully preserving the ability of the network. Ashok *et al.* [1] shrank a large teacher network to a small student network by removing redundant layers and decreasing the size of the remaining layers, where a reinforcement learning model was employed to learn the policy. He *et al.* [20] efficiently sampled the network architecture space by leveraging a reinforcement learning model, so that the model was compressed automatically without predefined pipelines. In this paper, we extend the deep reinforcement learning model to mine the channel-wise interactions for binary convolutional neural networks.

2.3 Hierarchical Reinforcement Learning

Discovering effective hierarchies of policies is a long standing research problem in reinforcement learning [14], [31], [44], [54], [57], [59] as the search space significantly shrinks with the enhancement of the search efficiency. Sutton *et al.* [54] proposed the options framework consisting of an agent in the top level that set a sub-goal, and a low-level agent that selected the primitive actions following the sub-goals. Kulkarni *et al.* [31] proposed hierarchical-DQN that integrated hierarchical action-value functions to operate at different time-scales. Vezhnevets *et al.* [57] replaced the pre-defined sub-goals with meaningful and explicit goals discovered by the top-level agent, and the adaptive sub-goals enhanced the performance on ATARI. [44] presented to use the off-policy experience to train the agents with high sample efficiency. Wang *et al.* [59] modeled the video caption as a two-level search problem, where the high-level Manager module designed the context of each segment and the low-level Worker generated the segment word by word following the guidance. To the best of our knowledge, we are the first to employ hierarchical reinforcement learning for network compression.

2.4 Feature Denoising

Feature denoising has been widely studied in visual analysis [12], [27], [58], [63] and speech processing [15], [25] due to the demands of learning robust features. The feature denoising methods can be categorized into two types: statistics based and learning based. For the former, elements of features are usually smoothed by the global or local statistics. Xie *et al.* [63] applied the smoothing filters including means and medians to eliminate the noise in the intermediate feature maps of convolutional neural networks, so that the networks were robust to adversarial attack. In the latter regard, regularization and denoising AutoEncoders are often applied to learn the robust feature. Vincent *et al.* [58] and Du *et al.* [12] employed the denoising AutoEncoders to learn image features that are invariant to various noise corruption in an unsupervised manner. Lu *et al.* [40] performed feature detection, identification and connection with L_1-norm regularization to preserve informative features, so that vertices of the mesh were updated with feature-awareness to eliminate the

noise. We employ the statistics based method in denoising interacted bitcount due to its simplicity in computation.

3 Channel-Wise Interaction Based Binary Convolutional Neural Networks

In this section, we first introduce neural networks with binary weights and activations briefly, which is efficient but suffers 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 in binary convolutional neural networks.

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 lth convolutional layer in a given L-layer CNN model, where (w_f^l, h_f^l) and (w_a^l, h_a^l) represent the width and height of filters and feature maps respectively in the lth 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 [50] 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 lth 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 for given receptive field. $sign$ means the sign function which maps number larger than zero 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:

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

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 solution $\boldsymbol{W_b^l} = sign(\boldsymbol{W_r^l})$ and $\boldsymbol{A_b^{l-1}} = sign(\boldsymbol{A_r^{l-1}})$ by assuming

$$sign(\boldsymbol{A_r^l}) \approx sign(sign(\boldsymbol{W_r^l}) \odot sign(\boldsymbol{A_r^{l-1}})).$$

However, due to the quantization error occurred in xnor and bitcount operations, the assumption does not always hold as shown in Fig. 1. The approximate solution has inconsistent signs in $\boldsymbol{A_b^l}$ compared with $sign(\boldsymbol{A_r^l})$, which is far from the optimal solution of (1). Moreover, the error is accumulated across layers and causes severe information loss in 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 $\boldsymbol{A_b^l}$.

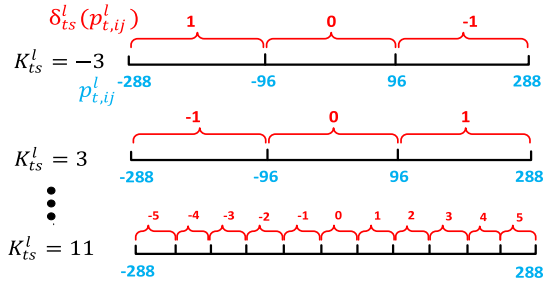


Fig. 3. Calculation of the interacted bitcount. The value range of teacher feature maps are partitioned into $|K_{ts}^l|$ intervals with equal length (blue) when considering its interaction (red) to student feature maps. The teacher and student feature maps are positively correlated if K_{ts}^l is positive and vice versa. When $|K_{ts}^l|$ is larger, the teacher feature map influences the student one more significantly. In this example, $N_0 = 288$ and U_0 is set as 0.001 so that the unit pixel modification is $[U_0 N_0] = 1$. (best viewed in color).

3.2 Interacted Bitcount

Applying xnor and bitcount operations brings significant quantization error, which leads to inconsistent signs in binary feature maps compared to their real-valued counterparts. Meanwhile, interdependencies among channels have been shown in many previous works [24], [47]. Feature response in one feature map can provide priors for that in its correlated feature map to alleviate inconsistent signs, which are termed as the teacher feature map and the student feature map respectively. The interacted bitcount adds a penalty to the conventional bitcount by considering the feature response in correlated channels

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

where the upscript l represents the variable in the lth convolutional layer. $p_{s,ij}^l$ and $p_{t,ij}^l$ are the feature response obtained by original bitcount, which are located in the ith row and jth column of the student and teacher feature map respectively. δ_{ts}^l is a penalty term providing priors for the student feature map based on the teacher one, and modifies $p_{s,ij}^l$ to $\widetilde{p}_{s,ij}^l$ as the output of interacted bitcount.

To prevent the network suffering from heavy computation overhead of interacted bitcount, we simply design δ_{ts}^l as an integer stair function parameterized by K_{ts}^l. Fig. 3 demonstrates the calculation of the interacted bitcount. We partition the value range of teacher feature maps into $|K_{ts}^l|$ intervals with equal length (blue) when considering its interaction (red) to student feature maps. The teacher and student feature maps are positively correlated if K_{ts}^l is positive and otherwise are negatively related. K_{ts}^l is set to be an odd integer so that no priors are provided if $p_{t,ij}^l$ is near zero without sufficient information. When the norm of K_{ts}^l, i.e., $|K_{ts}^l|$, is larger, the teacher feature map influences the student one more significantly in the lth layer. The output of δ_{ts}^l is obtained as follows:

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

where p_k is the origin of the kth interval in value range partition of the teacher feature map. N_0 is the maximum in value range of the teacher feature map. U_0 means a hyperparameter that decides the unit pixel modification. $[U_0 N_0]$ stands for the minimal integer larger than $U_0 N_0$.

3.3 Channel-Wise Interaction Mining via Policy Gradient

Since searching for the optimal channel-wise interactions is an NP-hard problem, we employ the policy gradient method to reduce the complexity. The policy $\pi_\theta(a|s)$ is parameterized by policy networks, where θ, s and a are the parameters of policy networks, states and actions respectively. The policy decides the optimal action to achieve the goal of search when given the state. The objective of policy gradient learning is to maximize the expected reward over the entire searching process.

The channel-wise interactions are defined as edges in the graph among channels, which are expressed as existence and influence. Existence of an edge depicts the correlation between two nodes, represented by one if their correlation is significant and zero otherwise. An edge's influence means the impact strength on the end node imposed by the start node. As partitioning the value range of teacher feature maps into more intervals stands for greater impact of the channel-wise interactions, we demonstrate the influence by K_{ts}^l. Mining channel-wise interactions is viewed as a Markov Decision Process (MDP). At each step, the agent takes an action to create edges, delete edges or keep edges unchanged to modify the edge existence in the graph, and assigns 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 $\mathcal{S}_f^l$ across layer index l

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

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 means the number of the channels in the lth layer. For the element $w_{es,ts}^l$ in the tth row and sth column of W_{es}^l, it equals to one if the directed interaction from tth to sth channel exists and zero otherwise. The influence space $\mathcal{S}_f^l$ is modeled by the influence matrix W_{fs}^l with odd integers. We limit the space with finite discrete numbers as $W_{fs}^l \in \{\pm 3, \pm 5, \pm 7, \ldots, \pm(2K_0+1)\}^{c^l \times c^l}$, where K_0 is a hyperparameter representing the size of action space. In our implementation, element $w_{fs,ts}^l$ in the tth row and sth column of W_{fs}^l is scaled to $sign(w_{fs,ts}^l) \cdot \frac{|w_{fs,ts}^l|-1}{2K_0}$ for normalization.

Action. The action set $\mathcal{A}$ is the direct product of action space in existence $\mathcal{A}_e^l$ and in influence $\mathcal{A}_f^l$ 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 $\{unchanged\}$ for remaining the existence invariant. $\mathcal{A}_f^l$ depicts all possible odd integers in W_{fs}^l for existing edges. Moreover, we stop to update the policy when the graph converges or achieving the maximal steps. The whole action set is described as

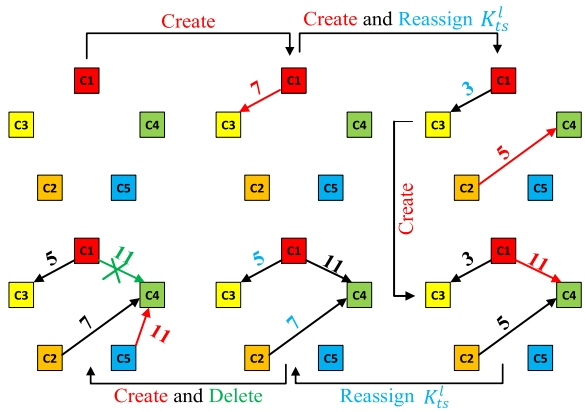


Fig. 4. An example for graph mining. We create edges, reassign K_{ts}^l and delete edges between different channels until finalizing the graph structure (best viewed in color).

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

Fig. 4 represents an example of stage transitions with actions. $W_{ea}^l \in [0, 1]^{c^l \times c^l}$ parameterizes the probability of actions that creating edges for existence, whose element $w_{ea,ij}^l$ in the ith row and jth column demonstrates the probability of connecting the directed edge from the ith channel to the jth one with the normalization $||W_{ea}^l||_1 = 1$. We select actions according to the following rules:

1) *Create*: The density of the existence matrix is defined as the ratio of ones in the existence matrix. When the density of existence matrix is sparser than the hyperparameter ρ_{max}, we create an edge directing to the jth channel from the ith one if the sampling strategy based on W_{ea}^l selects the element $w_{ea,ij}^l$ and the edge has not been connected.
2) *Delete*: The probability of deletion is formulated as $W_{ea}^{'l} = Norm(\{-\log w_{ea,ij}^l\}_{c_l \times c_l})$, where $Norm$ means the normalization operation that ensures $||W_{ea}^{'l}||_1 = 1$. The probability of edge creation and deletion is negatively correlated, because the low probability to connect an edge means high probability to keep it disconnected. Since differences of low probabilities in W_{ea}^l are very small and can only be revealed by their power exponent, we apply logarithm to represent the possibility of deletion. We delete the existing edge between the ith channel to the jth one if the sampling strategy chooses the element $w_{ea,ij}^{'l}$ in the ith row and jth column of $W_{ea}^{'l}$.
3) *Unchanged*: The existence of all edges keeps unchanged if no edges are created or deleted.

$W_{fa}^l \in [-1, 1]^{c^l \times c^l}$ parameterizes the delta distribution of K_{ts}^l for actions of edge influence, and we select K_{ts}^l deterministically according to the element $w_{fa,ts}^l$ in the ith row and jth column of W_{fa}^l

$$K_{ts}^l = sign(w_{fa,ts}^l) \cdot [2 * [|K_0 w_{fa,ts}^l|] + 1]. \quad (4)$$

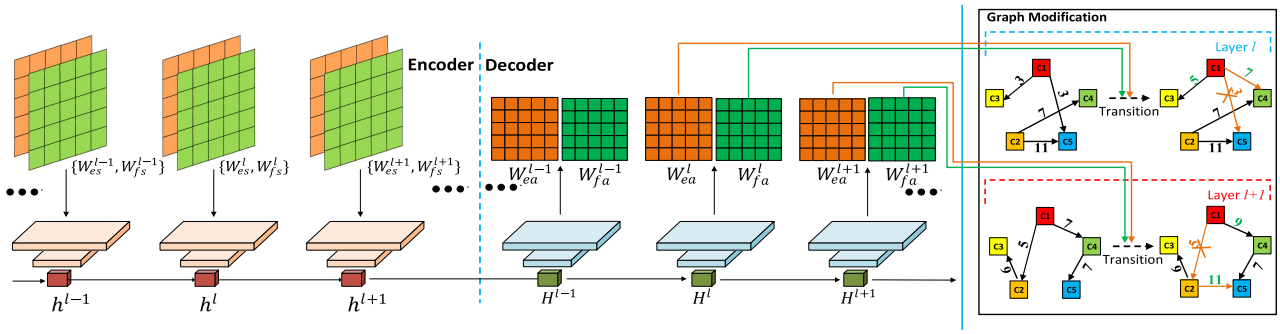


Fig. 5. Overall framework for training the CI-BCNN. The left part is the policy network which consists of encoders and decoders. The encoders take the state of each layer as input and decoders output the corresponding action probability matrix W_{ea}^{l} and W_{fa}^{l} for edge existence and influence according to the hidden variables. The right part stands for the graphs representing channel-wise interactions in binary convolutional neural networks, where the convolutional layer modifies its graph structure based on the action probability matrix (best viewed in color).

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 in round τ is modeled as follows:

$$\begin{aligned} r(s_\tau, a_\tau, s_{\tau+1}) &= r_c(s_\tau, a_\tau, s_{\tau+1}) + r_p(s_\tau, a_\tau, s_{\tau+1}) \\ &= \frac{1}{2}(sign(|C(s_\tau) - C(s_{\tau+1})| - h) + 1) \cdot sign(C(s_\tau) - C(s_{\tau+1})) \\ &+ \frac{1}{N}\sum_{l=1}^{L}\sum_{t,s}\sum_{i,j} sign(|p_{t,ij}^{l}(s_{\tau+1})| - |p_{s,ij}^{l}(s_{\tau+1})|), \end{aligned} \tag{5}$$

where $C(s_\tau)$ represents the cross-entropy loss of the binary neural networks with the graph mined in round τ, and h is a positive hyperparameter. $p_{t,ij}^{l}(s_{\tau+1})$ and $p_{s,ij}^{l}(s_{\tau+1})$ mean the pixel values in the ith row and jth column of the teacher and student feature maps, which are calculated by the interacted bitcount with the graph mined in round $\tau+1$. N stands for the number of total pixels of feature maps in the binary convolutional neural networks, which equals to $\sum_{l=1}^{L}\sum_{t,s}\sum_{i,j} 1$.

r_c encourages the graph imposed on the binary neural networks 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 to provide priors that correct inconsistent signs. Pixels representing what the convolutional filters aim to extract or opposite to that are usually positively or negatively activated respectively, and they carry more information. As a result, we encourage the L1 norm of activations in the teacher feature maps to be larger than that in the student feature maps.

We employ an Encoder-decoder RNN for the policy network, which takes the current state of graph W_{es}^{l} and W_{fs}^{l} as input, while outputs the action probability matrix W_{ea}^{l} and W_{fa}^{l}. Fig. 5 shows the overall framework for training our CI-BCNN with the policy network. The objective of the policy network is to maximize the expected reward over the entire searching process. As the reward is non-differentiable, we employ the REINFORCE [62] algorithm to directly obtain the expected gradients of policy networks for optimization (formulated in the supplementary material), which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.2988262.

4 Hierarchical Channel-Wise Interaction Based Binary Convolutional Neural Networks

We first propose channel-wise interaction mining via hierarchical policy gradient, and then present the denoising interacted bitcount for smoothing the channel-wise priors.

4.1 Channel-Wise Interaction Mining via Hierarchical Policy Gradient

The channel-wise interactions are sparse in binary convolutional neural networks as the independencies among channels are required to maintain the representational capacity. While CI-BCNN searches for the channel-wise interactions in large policy space where each channel can be correlated with others, the search deficiency obstacles the agent to obtain the optimal policy to represent the channel-wise interactions. In order to address these limitations, we further propose hierarchical channel-wise interaction based binary convolutional neural networks, where the correlations among channels are learned via hierarchical policy gradient.

Hierarchical reinforcement learning is a widely used technique that has been proven to be effective in conventional search tasks such as ATARI and Minecraft [55], [57], because it shrinks the search space and improves the search efficiency. Hierarchical policy gradient decomposes the policy space into two levels and searches for the optimal policy in each decomposed space respectively. The combination of policies in two levels yields the policy for the original search problem. There is a high-level agent (manager) that sets sub-goals and a low-level agent (worker) that provides sub-policy to accomplish the sub-goals. Compared with the conventional HRL setting, we take the extrinsic rewards obtained with given sub-goals as intrinsic rewards to train the worker. As the correlation among channels is sparse, mining channel-wise interactions can be accomplished effectively and efficiently in a two-level hierarchy: (1) the manager partitions the channels according to their correlation, which is the sub-goal for the worker; (2) the worker selects sub-policies to mine fine-grained channel-wise interactions for each partition guided by the sub-goal, and the combination of all sub-policies represents the global channel-wise interactions in a

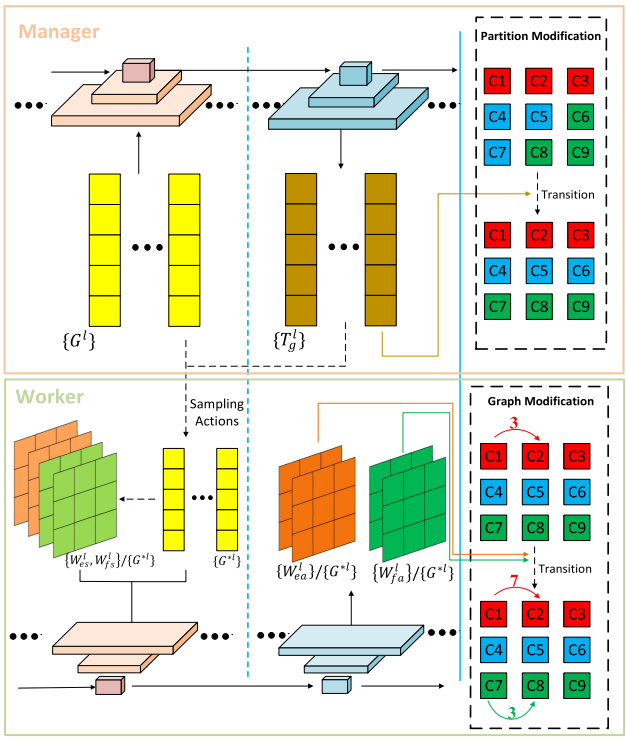


Fig. 6. The hierarchical policy network based on the Encoder-decoder RNN framework. Each recurrent module consists of a top-level agent named Manager and a low-level agent called Worker. The manager outputs the action probability for each channel group according to the former partitions. After sampling the action to modify partitions, the new group assignment is set as the sub-goal for the worker. The partition and former fine-grained channel-wise interactions in each group are input to the worker, and the action probability for rectifying the fine-grained interaction graph is output.

convolutional layer. Fig. 6 illustrates the framework of channel-wise interaction mining via hierarchical policy gradient. Similar to the CI-BCNN method, the policy network is designed with an Encoder-decoder RNN framework, where each recurrent module consists of a manager and a worker instead of a single agent. We first introduce the manager and then explain the worker in detail. Finally, we illustrate the joint training of the manager and the worker.

Manager. The manager aims to partition channels into different groups according to their correlation in a coarse manner, where positively or negatively related channels are assigned to one group and channels in different partitions are independent. Similar to the above CI-BCNN, we model the channel partition as a Markov decision process and train the agent with policy gradient. As the rewards are defined in the same form with that in CI-BCNN, here we only focus on states and actions.

The state space G includes the current partition for all the convolutional layers, which is represented by direct product of binary masks $\boldsymbol{g}_i^l \in \{0,1\}^{c^l \times 1}$ across the layer index l and the partition index i in each layer

$$G = \prod_{l=1}^{L} \prod_{i=1}^{n^l} \boldsymbol{g}_i^l \qquad s.t. \quad ||\boldsymbol{g}_i^l||_1 = k^l, \tag{6}$$

where n^l means the number of groups in the lth layer and k^l is a hyperparameter no more than $\frac{c^l}{n^l}$. The jth bit in the $\boldsymbol{g}_i^l$ is denoted as g_{ij}^l, which equals to one if the jth channel belongs to the ith group in the lth layer and equals to zero otherwise. As the shape of the policy network in the worker is fixed and regular, we set the number of channels in each group as a constant k^l via the L_1 norm constraint.

The action set $\mathcal{A}_g$ is also the direct product of partition modification actions $\boldsymbol{a}_{g,i}^l = \{1, 0, -1\}^{c^l \times 1}$ over all groups across layers, where l and i represent the index of the layer and the group. The jth element in $\boldsymbol{a}_{g,i}^l$ is represented as $a_{g,ij}^l$, which equals to one if the manager takes actions to add the jth channel to the ith group and equals to minus one if the jth channel is removed from the ith group. The correlation between the channel and the group remains unchanged if the corresponding element is zero. For each partition, the manager can select actions with two choices: (1) adding a channel and removing a channel. In order to satisfy the constraint in 6, channel removal and addition for each group are simultaneous; (2) Remaining the partition unchanged. $\boldsymbol{t}_{g,i}^l \in [0,1]^{c^l \times 1}$ parameterizes the channel addition probability for the ith group in the lth layer, and $T_g^l = \prod_{i=1}^{n_l} \boldsymbol{t}_{g,i}^l$ means the overall probabilities in the lth layer. The jth element $t_{g,ij}^l$ stands for the probability of the jth channel to be added to the ith channel with the normalization $\sum_j t_{g,ij}^l = 1$. Actions are selected from two possible choices according to the following criterions:

1) *Add a channel and remove a channel:* The partition modification confidence is defined as the largest element divided by the second largest one in $\boldsymbol{t}_{g,i}^l$. When the partition modification confidence is higher than a hyperparameter p_0, the jth channel is added to the ith group if the sampling strategy chooses the element $t_{g,ij}^l$ and the selected jth channel does not belong to the ith partition before addition. When a channel is added to the ith group, the channel with minimal $t_{g,ij}^l$ in the ith group should be removed.
2) *Unchanged:* The partition is not modified if no addition and removal actions are taken.

Worker. The worker mines the fine-grained channel-wise interactions for each partition set by the manager. Similar to CI-BCNN, the worker takes the current graph structure of each group across all layers as the input and outputs the action probability including edge existence and influence for each group to modify the graph. States, actions and rewards are the same with those in CI-BCNN despite the following differences: (1) The search space is a subset of the original space, where channels in one partition forms a new sub-graph to represent the fine-grained channel-wise interactions. As a result, the size of the matrix to illustrate states and actions is $n_l \times \frac{c_l}{n_l} \times \frac{c_l}{n_l}$ for the worker in the lth layer. We denote the corresponding space as $\{\mathcal{S}_e^l, \mathcal{S}_f^l\}/\{G^{*l}\}$ and $\{\mathcal{A}_e^l, \mathcal{A}_f^l\}/\{G^{*l}\}$, which means the original space is decomposed by G^{*l} and G^{*l} is the channel partition given by the manager; (2) Besides the influence matrix and the existence matrix are input to the policy network, the binary mask $\boldsymbol{g}_i^l$ to represent partition is also fed into the policy network as sub-goals set by the manager. (3) The state and the channel assignment of all partitions form a input mini-batch, through which we train the policy network of the worker.

Similar to CI-BCNN, the manager stops to reassign channels to different groups and the worker does not modify the

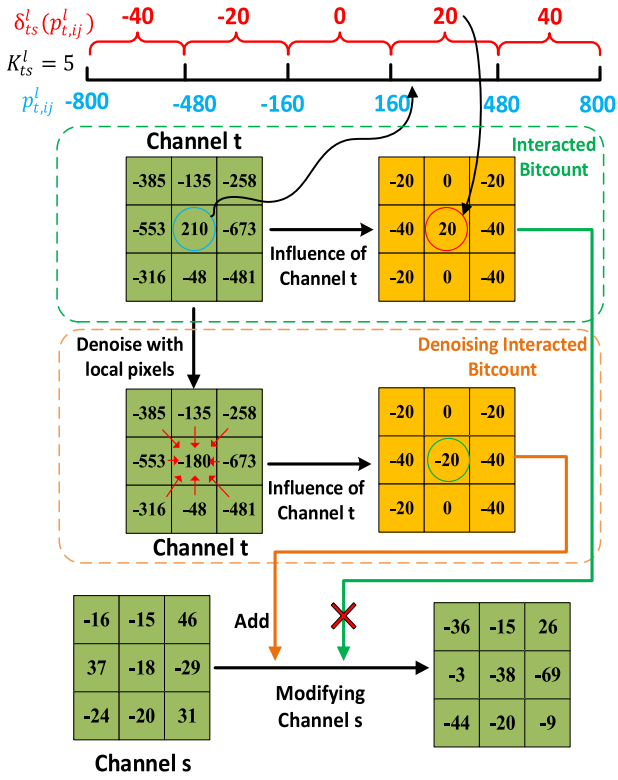


Fig. 7. Comparison between the presented IB and the DIB. IB yields the channel-wise priors of the t_{th} channel to the s_{th} channel pixel by pixel, the modification to the s_{th} channel may rot the original information due to the noise in t_{th} channel. DIB employs the denoising operation based on the local pixels of teacher feature maps to eliminate the noise in the central pixel, and produces channel-wise priors with robust instructions. For example, the central pixel in the s_{th} channel is modified from -18 to $-38\ (=-18-20)$ in DIB and to 2 $(=-18+20)$ in IB.

graph that represents channel-wise interactions until their convergence or achieving the maximal steps. The manager and the worker are trained alternatively with the following assumption: The manager is well-posed when training the worker as we can disable the sub-goal exploration and only update the policy for each partition. The worker proposes Oracle policy when training the manager as we freeze the sub-policy update and explore partitions. In this way, the original channel-wise interaction search space is decomposed with significant shrinkage, so that the search efficiency is improved and the optimal policy is selected.

4.2 Denoising Interacted Bitcount

When imposing the mined channel-wise interactions pixel by pixel on the student feature maps via interacted bitcount, noise in the teacher feature maps corrupts the information in the student ones and decreases the accuracy significantly. In order to alleviate noise in channel-wise priors, we further propose denoising interacted bitcount. Fig. 7 demonstrates the difference between IB and DIB. Teacher feature maps provide channel-wise interactions with feature denoising techniques, where noisy pixels can be smoothed through the priors from their local region. The denoising interacted bitcount rectifies the original interacted bitcount in the following form:

$$\widetilde{p}^l_{s,ij} = p^l_{s,ij} + \sum_t \delta^l_{ts}(y^l_{t,ij}), \tag{7}$$

where $y^l_{t,ij}$ is the denoised pixel of $p^l_{t,ij}$ and we apply different feature denoising techniques to obtain $y^l_{t,ij}$. Since the computational efficiency during the inference phase is required, we employ simple operations for DIB and alleviate the noise according to other pixels in the local region. We experiment with the following instantiations of the denoising operations in our DIB.

Means. Applying the mean of the pixels p_Ω in the defined local region to represent the central pixel is natural and simple, which is written as follows:

$$y^l_{t,ij} = \frac{1}{|\Omega(p^l_{t,ij})|} \sum_{p_\Omega \in \Omega(p^l_{t,ij})} p_\Omega, \tag{8}$$

where $\Omega(p^l_{t,ij})$ means the local region of $p^l_{t,ij}$ and $|\Omega(p^l_{t,ij})|$ represents the number of pixels in $\Omega(p^l_{t,ij})$. Since the division for calculating local means is time-consuming and inconsistent with the binary convolution, we propose an alternative algorithm to approximate the local means as follows:

$$y^l_{t,ij} = p^l_{t,ij} + \Delta_0 \cdot \sum_{p_\Omega \in \Omega(p^l_{t,ij})} sign(p_\Omega - p^l_{t,ij}), \tag{9}$$

where Δ_0 is an integer hyperparameter to control the effect of local pixels on the central pixel. The pixel $p^l_{t,ij}$ is penalized by Δ_0 positively if the local pixel is larger than the central one and vice versa.

Medians. Local medians can also be utilized in feature denoising, which is formulated as follows:

$$y^l_{t,ij} = \{y | y = median(p_\Omega), p_\Omega \in \Omega(p^l_{t,ij})\}, \tag{10}$$

where the median is over the local region. Medians are well known to be helpful to eliminate salt-and-pepper noise and remove outliers of the similar kind. As the median of local region can be found via merge-sort algorithms, the computational cost is only $\mathcal{O}(\log|\Omega(p^l_{t,ij})|)$.

Modes. Modes demonstrate the value that occurs most frequently. As the pixels on the local region of teacher feature maps are usually varied, we consider the mode of the penalty term to obtain the denoised pixel, which is formulated as follows:

$$y^l_{t,ij} = \{y | \delta^l_{ts}(y) = mode(\delta^l_{ts}(p_\Omega)), p_\Omega \in \Omega(p^l_{t,ij})\}, \tag{11}$$

$y^l_{t,ij}$ can be any value providing the penalty that is the mode of all penalties in $\Omega(p^l_{t,ij})$ for the student feature map. For the simplicity of DIB, we modify (7) to the following form instead of calculating the value of $y^l_{t,ij}$:

$$\widetilde{p}^l_{s,ij} = p^l_{s,ij} + \sum_t mode(\delta^l_{ts}(p_\Omega)), \tag{12}$$

where $p_\Omega \in \Omega(p^l_{t,ij})$ and we randomly select a candidate if there are multiple modes in the local region.

In order to keep the efficiency of binary convolutional neural networks, we define the local region with suitable size (e.g, 3×3) which balances the richness of local information and the computational cost.

5 Experiments

In this section, we evaluate the proposed CI-BCNN and HCI-BCNN on two datasets for image classification: CIFAR-10 and ImageNet. First, we introduce the implementation details, and illustrate the effectiveness and intuitive logic of channel-wise interactions by toy examples. Second, we investigate the influence of hyperparameters and the proposed techniques of the presented method by ablation study. Third, we compare the CI-BCNN and HCI-BCNN with state-of-the-art binarized neural networks regarding the accuracy. Finally, we analyze the storage and computation complexity during inference in comparison with other methods.

5.1 Implementation Details

For both CI-BCNN and HCI-BCNN, we trained the backbone with the VGG-small [68] and ResNet20 architectures on the CIFAR-10 dataset. We employed ResNet18 and ResNet34 for the backbone in the experiments on the ImageNet dataset. We iteratively trained the binary neural networks and the agent for mining channel-wise interactions. 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 [50] 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 one epoch to absorb the scaling factors. In the inference stage, the scaling factors were removed for further acceleration.

The training of policy network was different in CI-BCNN and HCI-BCNN. For CI-BCNN, we applied two convolutional layers with a fully-connected layer for the encoder and employed 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 lth layer for memory saving and computation acceleration. For the training of the policy network, we set the hyperparameters U_0, ρ_{max}, K_0 and α as 0.01, 0.1, 2 and 0.001 respectively in the experiments compared with the state-of-the-art methods. For HCI-BCNN, we employed the same architecture used in the policy network of CI-BCNN for the worker with the only difference that we added a fully-connected branch in the encoder to feed forward the channel assignment for partitioning. We used two fully-connected layers in multi-branches for the encoder and decoder in each module of the manager. For most experiments, we set the number of partitions to $n_l = 4$. p_0 equaled to 1 in the first epoch and was changed to 1.2 in the following epoches. Other hyperparameter settings were the same with the CI-BCNN. For the proposed DIB, the 3×3 local region and the medians were adopted in most experiments, and Δ_0 in (9) was set as 10 when applying the means in DIB.

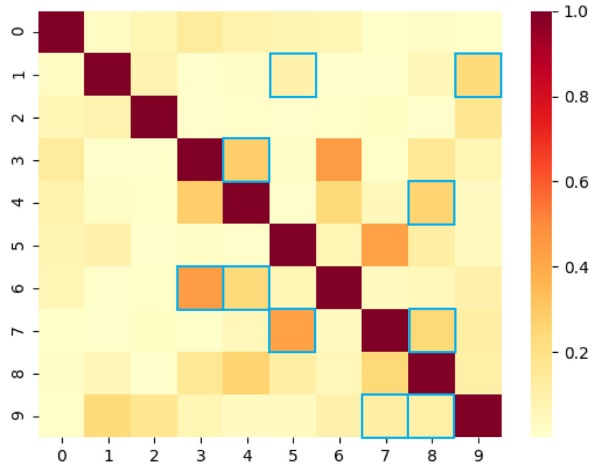


Fig. 8. The square of correlation coefficients among 10 channels in the binary convolutional layer. Darker colors represent higher correlations and the blue box demonstrates the connections mined by our policy gradient networks.

5.2 Toy Examples

The thought of the CI-BCNN and HCI-BCNN is to effectively search for the correlational graph among channels to correct inconsistent signs in binary feature maps caused by the xnor and bitcount operations. We conducted simple experiments to show the correctness of our thoughts with intuition.

Effectiveness of Channel-Wise Interactions. 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 and evaluated the network on MNIST [32]. Fig. 8 shows the square of correlation coefficients among different channels in the binary convolutional layer, where darker colors mean higher correlations. In this figure, the horizontal axis represents the student feature maps while the vertical axis means the teacher feature maps. The blue box represents the channel-wise interactions learned by our policy network. As can be seen, the learned channel-wise interactions are reliable since the channel pairs with high correlation are found.

Effectiveness of Interacted Bitcount. Our interacted bitcount utilized channel-wise interactions to provide priors to correct inconsistent signs in the binary feature map. We expect that more pixels in binary feature maps have the same signs with their full-precision counterparts, so that information of input images is preserved during inference. We conducted experiments on CIFAR-10 with the VGG16 architecture. 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 layers in the top generally obtain more inconsistent signs due to the quantization error accumulation in the forward propagation. Generally speaking, our CI-BCNN increases the sign consistency of pixels in intermediate feature maps, which is benefited from the priors provided by interacted bitcount.

TABLE 1
Comparison on the Ratio of Pixels With Consistent Signs in Different Layers and Corresponding Accuracies of CI-BCNN

Layer Index	C1_2	C2_1	C2_2	C3_1	C3_2	C3_3	C4_1	C4_2	C4_3	C5_1	C5_2	C5_3	Acc.(%)
Bitcount	0.727	0.734	0.704	0.722	0.678	0.688	0.659	0.612	0.634	0.586	0.535	0.505	90.23
Interacted Bitcount	0.751	0.752	0.744	0.713	0.695	0.674	0.660	0.626	0.623	0.616	0.542	0.525	92.47

5.3 Performance Analysis

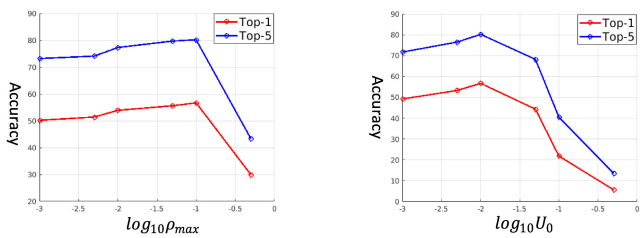
In this section, we analyze the effect of hyperparameters and search methods in CI-BCNN at first, and then show the effectiveness of the proposed techniques in HCI-BCNN and the influence of corresponding hyperparameters via ablation study.

5.3.1 Ablation Study for CI-BCNN

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_{max}, ratios of unit pixel modification U_0 and the search methods. We report the classification top-1 and top-5 accuracies on the ImageNet dataset with the ResNet18 architecture.

Performance w.r.t. the Maximal Density of Existence Matrix ρ_{max}. The maximal density of existence matrix ρ_{max} is defined as the maximal proportion of ones in the matrix, which is positively correlated with the final density of existence matrix. 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 Fig. 9a. Medium density provides priors that correct inconsistent signs in intermediate feature maps. High density connects the channels with weak correlation in the graph. Low density fails to consider priors, which is unable to alleviate inconsistent signs in binary feature maps caused by xnor and bitcount operations.

Performance w.r.t. the Ratio of Unit Pixel Modification U_0. Larger U_0 in the interacted bitcount stands for more significant modification, so that the channel-wise priors become more important when compared with the posterior information gained from input images. Fig. 9b shows the performance versus different U_0. Medium U_0 achieves the best performance. 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.



(a) Varying ρ_{max} (b) Varying U_0

Fig. 9. 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.

Performance w.r.t. the Search Methods. Employing effective search methods to obtain the optimal channel-wise interaction graph is crucial for learning binary neural networks with decreased quantization error. Table 2 demonstrates the accuracies of binary convolutional neural networks on ImageNet without graph, with completely-connected graph, with graph obtained via random search and with the proposed CI-BCNN, where we only search for the edge influence in the setting of completely-connected graph. The lack of graph fails to solve the information loss caused by inconsistent signs, while the completely-connected graph leads to redundancy of the feature maps and harms the representational power of the networks. Moreover, the random search is inefficient due to the search deficiency. We also plot the reward averaged over five random seeds in the supplementary material. available online, which proves that the policy network in CI-BCNN learns to optimize the right objectives.

5.3.2 Ablation Study for HCI-BCNN

As the hierarchical channel-wise interaction learning decomposes the search process into channel partition and fine-grained interaction mining, the search efficiency is significantly improved and the optimal policy is obtained. To verify the importance of hierarchical reinforcement learning, we implemented HCI-BCNN with different numbers of partitions for each layer. Meanwhile, since the noise in the teacher feature maps can be smoothed by denoising interacted bitcount, we also trained our HCI-BCNN with different combinations of denoising operations and sizes of local regions. We adopted ResNet18 as the backbone and employed ImageNet as the dataset for evaluation in the ablation study for HCI-BCNN. Top-1 and top-5 accuracies are reported in Table 3. Based on the results, we evaluate the influence of the proposed techniques and different hyperparameters on the final accuracy.

1) Comparing the accuracy obtained with different numbers of partitions, we know that decomposing the policy search into channel partition and fine-grained channel-wise interaction mining can enhance the performance of binary convolutional neural networks due to the increase of search efficiency. However, excess partitions shrink the search space with

TABLE 2
Top-1 and Top-5 Accuracies (%) on ImageNet of the CI-BCNN in the Architecture of ResNet18 With Different Search Strategies

Methods	Top-1	Top-5
No graph	51.08	72.59
Completely-connected graph	2.95	5.03
Random search	52.15	74.66
CI-BCNN	56.73	80.12

TABLE 3
Top-1 and top-5 Accuracies (%) w.r.t. Different Combinations of the Number of Partitions, Denoising Operations, and Sizes of Local Regions

Local region	Denoising operation	$n_l = 1$		$n_l = 2$		$n_l = 4$		$n_l = 8$		$n_l = 16$	
		top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
1×1	Mean/Median/Mode	56.73	80.12	56.89	80.33	57.53	80.68	57.90	80.82	53.25	75.43
3×3	Mean	56.83	80.21	57.62	80.75	57.91	80.93	57.82	80.74	53.94	76.39
	Median	57.01	80.39	57.47	80.72	**58.15**	**81.53**	58.01	81.03	53.79	76.07
	Mode	56.05	79.03	56.95	80.41	57.75	80.84	57.51	80.63	53.32	75.52
5×5	Mean	57.05	80.45	57.01	80.35	57.38	80.59	57.59	80.69	51.97	73.56
	Median	57.11	80.53	57.22	80.51	57.35	80.52	57.88	80.84	52.30	74.07
	Mode	56.95	80.32	57.13	80.39	57.22	80.41	57.37	80.55	51.59	73.31

We set the number of partitions to $n_l = 2^k$ (k is an integer) as each partition can consist of equal channels in VGG16 and ResNet18. The size of local region was assigned to 1×1, 3×3 and 5×5 to keep the symmetry of the local region and avoid heavy computation overhead.

significantly insufficient exploration, which excludes the optimal channel-wise interactions in the policy space.

2) Comparing the performance achieved with different sizes of local regions, we conclude that feature denoising techniques improve the accuracy with suitable selection of local regions as noise in channel-wise priors is alleviated. However, the choice of 5×5 local region degrades the accuracy because denoising the central pixel with local pixels in such large region also corrupts original information especially in high frequency.
3) Different denoising operations achieve similar accuracy boost in most cases. For the 1×1 local region which means the absence of feature denoising, means, medians and modes are all the pixel value itself and the performance is always the same. For both 3×3 and 5×5 region, modes perform worst as the pixel values are diverse in the local region, so that applying modes to denoise the central pixel is usually biased. Medians obtain the best performance among the three proposed feature denoising techniques in most cases.

5.4 Comparison With State-of-the-Art Methods

In this section, we compare the performance of our CI-BCNN and HCI-BCNN with existing methods including BNN [28], BC [7], BWN [50], Xnor-Net [50], Bi-Real-Net [38], ABC-Net [36], LQ-Nets [68], SYQ [13] HWGQ [6] and TTQ [70] through various architectures in image classification tasks on the CIFAR-10 and ImageNet datasets.

TABLE 4
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
HCI-BCNN	1/1	**93.15**	**91.82**

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 compare the accuracies of VGG-small [68] and ResNet20 quantized by different methods. Table 4 shows the results. The comparison clearly indicates the proposed CI-BCNN and HCI-BCNN outperform 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 was randomly cropped for training from the resized image whose shorter side was 256. For inference, we employed the 224×224 center crop from images. As demonstrated in [38], 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 and HCI-BCNN. We compare our method with state-of-the-art quantized networks in ResNet18 and ResNet34 architectures and report top-1 and top-5 accuracies in Table 5, where (ext) means the proposed CI-BCNN and HCI-BCNN with extra shortcut. Bi-Real-Net achieves the outstanding performance among neural networks with binary weights and activations by adding shortcut connections. However, it 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. Our method preserves the information of input samples during inference through the interacted bitcount and the mined channel-wise interactions, and achieves the state-of-the-art performance. Moreover, CI-BCNN and HCI-BCNN obtain higher accuracies compared with HWGQ,

TABLE 5
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	–	–
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
CI-BCNN	1/1	56.73	80.12	62.41	84.35
HCI-BCNN	1/1	**58.15**	**81.53**	**63.51**	**85.02**
Bi-Real-Net	1/1	56.40	79.50	62.20	83.90
CI-BCNN (ext)	1/1	59.90	84.18	64.93	86.61
HCI-BCNN (ext)	1/1	**60.99**	**84.23**	**65.89**	**87.38**

TABLE 6
Comparison of Storage Cost and FLOPs of Different Methods With the ResNet18 and ResNet34 Architectures

		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
	HCI-BCNN	33.5Mbit	1.60×10^8
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
	HCI-BCNN	43.5Mbit	1.89×10^8

which employs two-bit activations. While the policy space is too large and the noise of teacher feature maps corrupts the original information of student feature maps in CI-BCNN, the proposed HCI-BCNN decomposes the search process to shrink the policy space and alleviate the noise in channel-wise interactions by denoising operations. As the channel-wise interactions are optimal and more robust in HCI-BCNN, it further boosts the top-1 accuracy by 1.42 percent (58.15-56.73) and 1.10 percent (63.51-62.41) with ResNet18 and ResNet34 respectively. In short, CI-BCNN and HCI-BCNN are more competitive than the state-of-the-art neural networks with binary weights and activations.

5.5 Complexity Analysis

We analyze the computational and storage complexity of CI-BCNN and HCI-BCNN 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 [17]. Because current generation of CPUs can implement 64 binary operations parallel in one block, the total FLOPs are calculated as the number of floating point multiplications plus $\frac{1}{64}$ of the amount of binary multiplications. The results are illustrated in Table 6 with our implementation settings.

The proposed CI-BCNN reduces 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 and HCI-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 of CI-BCNN is resulted from interacted bitcount, which is insignificant compared with standard binary convolutions. The computational overhead is slightly higher in HCI-BCNN than that in CI-BCNN due to the denoising operations in DIB, but the increase in FLOPs is still acceptable. On the contrary, our CI-BCNN and HCI-BCNN save 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 and HCI-BCNN require less memory usage and fewer FLOPs.

6 Conclusion

In this paper, we have proposed a channel-wise interaction based binary convolutional neural networks approach for efficient inference. Our 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. We have also proposed a hierarchical channel-wise interaction based binary convolutional neural networks approach to shrink the search space with improved efficiency. Moreover, we have presented denoising interacted bitcount to alleviate the noise in channel-wise priors. Extensive experimental results have demonstrated the effectiveness and the efficiency of the proposed approach.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61822603, Grant U1813218, Grant U1713214, and Grant 61672306, in part by the Beijing Academy of Artificial Intelligence (BAAI) under Grant BAAI2020ZJ0202, in part by a grant from the Institute for Guo Qiang, Tsinghua University, in part by the Shenzhen Fundamental Research Fund (SubjectArrangement) under Grant JCYJ20170412170602564, and in part by Tsinghua University Initiative Scientific Research Program. Parital of this work was presented in [60]. For more information, please visit: https://github.com/ZiweiWangTHU/CI-BCNN.git

References

[1] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2N learning: Network to network compression via policy gradient reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.

[2] V. Belagiannis, A. Farshad, and F. Galasso, "Adversarial network compression," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 431–449.

[3] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 459–468.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 850–865.

[5] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "Back to simplicity: How to train accurate BNNs from scratch?" pp. 1–9, 2019, *arXiv: 1906.08637*.

[6] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5406–5414.

[7] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[9] M. Denil *et al.*,"Predicting parameters in deep learning," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[10] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[11] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 1002–1014, Apr. 2018.

[12] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[13] J. Faraone, N. Fraser, M. Blott, and P. H. W. Leong, "SYQ: Learning symmetric quantization for efficient deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4300–4309.

[14] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," in *Int. Conf. Learn. Representations*, 2017, pp. 1–13.

[15] F. G. Germain, Q. Chen, and V. Koltun, "Speech denoising with deep feature losses," in *Proc. Interspeech*, 2019, pp. 2723–2727.

[16] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–13.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[19] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, vol. 2, pp. 1398–1406.

[20] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 784–800.

[21] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proc. 32nd Int. Conf. Int. Conf. Mach.*, 2015, pp. 597–606.

[22] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–10.

[23] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," pp. 1–9, 2017, *arXiv: 1704.04861*.

[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[25] B. Huang, D. Ke, H. Zheng, B. Xu, Y. Xu, and K. Su, "Multi-task learning deep neural networks for speech feature denoising," in *Proc. Annu. Int. Symp. Comput. Architecture*, 2015.

[26] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 105–114.

[27] W.-B. Huang *et al.*, "Building feature space of extreme learning machine with sparse denoising stacked-autoencoder," *Neurocomputing*, vol. 174, pp. 60–71, 2016.

[28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv , and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[29] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size," pp. 1–13, 2016, *arXiv:1602.07360*.

[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Univ. Toronto, vol. 1, no. 4, pp.1–7, 2009.

[31] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[33] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–12.

[34] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2181–2191.

[35] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.

[36] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 344–352.

[37] C. Liu *et al.*, "Circulant binary convolutional networks: Enhancing the performance of 1-bit DCNNs with circulant back propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2691–2699.

[38] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 722–737.

[39] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3290–3300.

[40] X. Lu, Z. Deng, and W. Chen, "A robust scheme for feature-preserving mesh denoising," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 3, pp. 1181–1194, Mar. 2016.

[41] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[42] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–13.

[43] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.

[44] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3303–3313.

[45] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4293–4302.

[46] A. Nech and I. Kemelmacher-Shlizerman , "Level playing field for million scale face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3406–3415.

[47] H. Peng, J. Wu, S. Chen, and J. Huang, "Collaborative channel pruning for deep networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5113–5122.

[48] A. Pirinen and C. Sminchisescu, "Deep reinforcement learning of region proposal networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6945–6954.

[49] Y. Rao, D. Lin, J. Lu, and J. Zhou, "Learning globally optimized object detector via policy gradient," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6190–6198.

[50] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[52] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, 2016, Art. no. 484.

[53] J. S. Supancic III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 322–331.

[54] R. S. Sutton, D. Precup, and S. P. Singh, "Intra-option learning about temporally abstract actions," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, vol. 98, pp. 556–564.

[55] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1553–1561.
[56] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–10.
[57] A. S. Vezhnevets *et al.*, "FeUdal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3540–3549.
[58] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[59] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Yang Wang , "Video captioning via hierarchical reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4213–4222.
[60] Z. Wang, J. Lu, C. Tao, J. Zhou, and Q. Tian, "Learning channel-wise interactions for binary convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 568–577.
[61] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–11.
[62] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.
[63] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 501–509.
[64] J. Yang *et al.*, "Neural aggregation network for video face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4362–4371.
[65] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7370–7379.
[66] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1349–1358.
[67] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep reinforcement learning for visual object tracking in videos," pp. 1–10, 2017, *arXiv: 1701.08936*.
[68] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 365–382.
[69] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016.
[70] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–10.
[71] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.

Ziwei Wang (Student Member, IEEE) received the BS degree from the Department of Physics, Tsinghua University, China, in 2018. He is currently working toward the PhD degree in the Department of Automation, Tsinghua University, China. His research interests include network compression, binary representation, and reinforcement learning.

Jiwen Lu (Senior Member, IEEE) received the BEng degree in mechanical engineering and the MEng degree in electrical engineering from the Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the PhD degree in electrical engineering from Nanyang Technological University, Singapore, in 2012. He is currently an associate professor with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include computer vision, pattern recognition, and machine learning. He has authored or coauthored more than 250 scientific papers in these areas, where more than 70 of them are the IEEE Transactions papers and 60 of them are CVPR/ICCV/ECCV/NeurIPS papers. He serves the co-editor-of-chief of the *Pattern Recognition Letters*, an associate editor of the *IEEE Transactions on Image Processing*, the *IEEE Transactions on Circuits and Systems for Video Technology*, the *IEEE Transactions on Biometrics, Behavior, and Identity Science*, and the *Pattern Recognition*. He was/is a member of the Multimedia Signal Processing Technical Committee and the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society, and a member of the Multimedia Systems and Applications Technical Committee and the Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems Society.

Jie Zhou (Senior Member, IEEE) received the BS and MS degrees both from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the PhD degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995. From then to 1997, he served as a postdoctoral fellow with the Department of Automation, Tsinghua University, Beijing, China. Since 2003, he has been a full professor with the Department of Automation, Tsinghua University, Beijing, China. His research interests include computer vision, pattern recognition, and image processing. In recent years, he has authored more than 100 papers in peer-reviewed journals and conferences. Among them, more than 30 papers have been published in top journals and conferences such as the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Image Processing*, and CVPR. He is an associate editor for the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and two other journals. He received the National Outstanding Youth Foundation of China Award.