IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 3, 2018, accepted March 13, 2018, date of publication March 23, 2018, date of current version April 23, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2818887

Rotation Invariant Local Binary Convolution Neural Networks

XIN ZHANG[1], YUXIANG XIE[1], JIE CHEN[2], (Member, IEEE), LINGDA WU[3], QIXIANG YE[4], (Senior Member, IEEE), AND LI LIU[1,2]

[1]College of Information System and Management, National University of Defense Technology, Changsha 410073, China
[2]Center for Machine Vision and Signal Analysis, University of Oulu, 90014 Oulu, Finland
[3]Space Engineering University, Beijing 101416, China
[4]University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Yuxiang Xie (yxxie@nudt.edu.cn)

This work supported in part by the National Natural Science Foundation of China under Grant 61571453 and Grant 61202336, in part by the Natural Science Foundation of Hunan Province under Grant 14JJ3010, and in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant 2017JJ1007.

ABSTRACT Convolutional neural networks (CNNs) have achieved unprecedented successes in computer vision fields, but they remain challenged by the problem about how to effectively process the orientation transformation of objects with fewer parameters. In this paper, we propose a new convolutional module, local binary orientation module (LBoM), which takes advantages of both local binary convolutional and active rotating filters to effectively deal with the rotation variations with fewer parameters. LBoM can be naturally inserted to popular CNN models and upgrade them to be rotation invariant local binary CNNs (RI-LBCNNs). RI-LBCNNs can be learned with off-the-shelf optimization approaches in an end-to-end manner and fulfill image classification tasks. Extensive experiments on four benchmarks show that RI-LBCNNs can perform image classification with fewer network parameters and significantly outperform the baseline LBCNN when processing images with large rotation variations.

INDEX TERMS Deep learning, rotation invariance, convolutional neural network, local binary filter.

I. INTRODUCTION

Recently, deep Convolutional Neural Networks (CNNs) achieve impressive results for image classification [1]–[7] and object detection [8]–[13]. There have emerged some well-known architectures, such as AlexNet [1], ZFNet [14], NIN [3], VGGNet [4], GoogLeNet [7] and ResNet [6], which push this bloom into new peaks increasingly. Plenty of experiments have validated that architectures based on CNNs with sufficient data provided, can achieve top tier performance on public benchmarks [1], [4]–[13], [15].

In general, this is true for CNNs with wealthy memory and sufficient disk storage. However, for some front-end devices (e.g., cell phones, unmanned aerial vehicles (UAVs), GoPro) with limited resources (i.e., memory and storage), CNNs with heavily learnable parameters are unsuitable. The networks need to make a compromise between efficiency and accuracy. In order to solve this problem, plenty of researches based on the compressed network are proposed. For example, Wu *et al.* [16] quantized the weights of both convolutional layers and fully-connected layers to compress the storage and computation cost of network. Juefei-Xu *et al.* [17] replaced the traditional convolutional layers with Local Binary Convolutional (LBConv) to make a compression for learnable parameters.

On the other hand, there has been longstanding interests in developing robust features invariance to rotation [18]–[24]. A challenge in real-world scenarios is that the object(s) in an image might show arbitrary orientations instead of being canonical orientation. For a deep CNN, the absence of rotation invariance property will result in the degradation of the classification performance confronting images with these rotated objects. The pre-defined max-pooling mechanism endows conventional CNNs [1], [4], [6], [7] the capacity of processing small transitions, including scale change and small rotation of the input image, but they still have limited capacities to classify the images with large pose variance, as showed in Fig. 1.

Orientation information encoding is an important procedure in the image classification pipeline [18], [22], [23]. In addition, rotation-invariant feature extraction is usually a complex process [18], being computationally demanding. To this end, we present a new architecture of CNN named

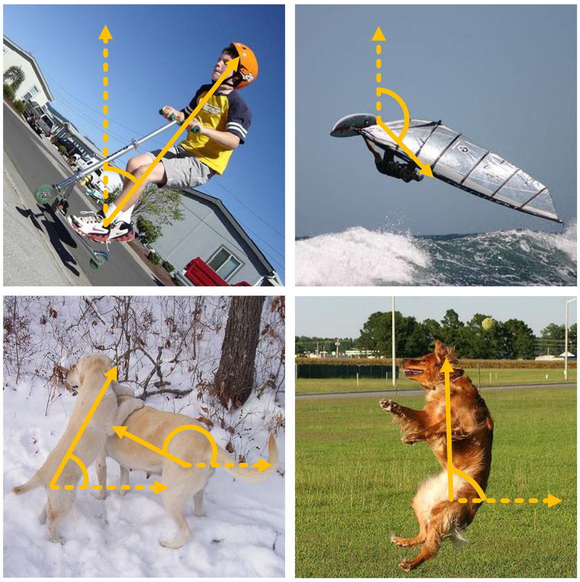

FIGURE 1. Illustration of some images with rotated objects. The dash line indicates the canonical orientation of an object, while the solid line is the true orientation. Deep CNNs need to classify images with objects in arbitrary orientations.

Rotation Invariant Local Binary Convolutional Neural Network (RI-LBCNN) to reduce learnable parameters and to endow CNNs the capability of orientation invariance, while achieving on-par performance with the state-of-the-art CNN architectures. Specifically, we introduce the Local Binary orientation Module (LBoM). Using this module, an Rotation Invariant Local Binary Convolutional Neural Network is proposed by inserting this module into a traditional CNN. An LBoM is composed of two components, i.e., a three-layer steerable module (two layers for the first component and one for the second one.), which takes advantages of both Local Binary Convolutional [17] and Active Rotating Filters [23]. Through replacing the convolutional layers in CNNs with LBoMs, RI-LBCNN can also be easily implemented. In addition, LBoM can be implanted to other advanced CNNs [4], [6], [7], learned in an end-to-end manner without additional modification to the learning process. In summary, the contributions of this paper are as follows: 1) We introduce the LBoM and propose RI-LBCNN based on LBoM, which endows CNNs with rotation invariance property and reduces the number of learnable parameters; 2) Popular CNN architectures (including VGGNet [4], ResNet [6] and WideResNet [5]) based on LBoM significantly outperform the state-of-the-art binary CNNs, for image classification with large rotation variances.

The short version of this work was published [25]. This extended version has significant improvement in the following aspects: 1) The ablation experiments are provided to explore the details of LBoM; 2) The experiments on Outex[1] texture dataset are provided to verify the generalization capability of the proposed RI-LBCNN; 3) Taking advantage of the promising experiments on several benchmarks based on RI-LBCNN, more detailed insights and analyses are provided; 4) We also examine the performance of different sparsity for RI-LBCNN.

The rest of this paper is organized as follows. Section II reviews the related literatures for rotation-invariant features and deep compression networks. Sections III describes the proposed LBoM and RI-LBCNN. In Section IV, we evaluate the performance of the proposed RI-LBCNN with extensive experiments on four popular datasets, and present the comparison with the state-of-the-art approaches. Section V discusses computation complexity and rotation invariance of the proposed LBoM. In Section VI, we conclude the paper and indicate the future work.

II. RELATED WORKS

In this section, we discuss the related works, covering hand-crafted rotation invariant features and modeling transformations based on CNNs. Meanwhile, we give a brief introduction to the evolution of compression techniques on neural networks.

A. ROTATION-INVARIANT FEATURES

1) HANDCRAFTED FEATURES

Challenging variations in real world consist of rotation, affine, scale, illumination, cluttering etc. Meanwhile, a simple way to tackle rotation variance in most computer vision researches is to use well-designed handcrafted features. The pre-defined features such as Scale Invariant Feature Transform (SIFT) [18], Rotation Invariant Local Binary Pattern (RI-LBP) [19], and Rotation Invariant Histogram of Gradients (RI-HoG) [20] are rotation invariant. With the dominant gradient orientation being aligned, SIFT achieved rotation invariance as well as the robustness to perspective transformations. Another representative domain-specific descriptor is scale and rotation invariant LBP [19]. Through assigning a minimum value to the cyclical original LBP, the rotation invariance is achieved. In addition, Fourier analysis in polar and spherical coordinates is employed to achieve rotation-invariant Histogram of Gradients [20].

These features perform well in many computer vision applications, but they have some limitations: 1) The designing of these hand-crafted features is time expensive and heavily depends on the experience; 2) Most of these well-designed features are domain-specific and can only tackle with specific transformation variances; 3) These mechanisms do not work well for arbitrary positions or dense feature computations.

2) DEEP NEURAL NETWORKS FEATURES

Generally, CNNs [1] consist of alternatively stacked convolutional layers and pooling layers, followed by several fully connected layers. These architectures have shown impressive performance on various fields, but they lack of abilities to be rotation invariance. To tackle global and arbitrary

[1] http://www.outex.oulu.fi/

rotation variance, data augmentation is usually used to achieve local/global invariance [26]. It improves the network performance through enlarging the volume of datasets with some pre-defined rules. TI-Pooling [22] uses images rotated in different angles as input to the parallel network architecture, which is followed by a pooling strategy at the first fully connected layer to learn the rotation-invariant feature.

Spatial Transform Network (STN) [21] shows some new hints to look for the robust representation of the input image. It learns the transformation matrix of the input image through an additional neural network framework. STN can be inserted into any existing CNN architectures to encapsulate with spatial invariance, but the problem on how to estimate the global transformation parameters precisely remains unsolved.

Zhou *et al.* [23] proposed to use the prior knowledge of rotation to the basic element in CNNs, i.e., the convolution operator. They introduced Actively Rotating Filters (ARFs) to generate feature maps which encode the location and orientation information [23]. Oriented Response Networks (ORNs) can be naturally fused with any popular deep learning architecture, as well as latest techniques (BatchNorm [27], ReLU [1]). Nevertheless, the number of learnable parameters in ORNs remains large despite the parameters reduction in the convolution layers. Focused on the multi-oriented pedestrian detection, Weng *et al.* [28] proposed a Global Polar Pooling (GP-Pooling) operator to capture rotational shifts in convolution features, which is further used to rectify the detection results. Marcos *et al.* [24] applied each convolution filter at different orientations and extracted a vector field from feature maps. Using the vector fields as the inputs, a rotation-invariant/equivariant architecture RotEqNet is achieved.

B. NETWORK COMPRESSION

CNNs are typically over-parametered, which thus are suffered from redundant parameters in their modules [29]. This leads to large model size in terms of both memory usage and disk space. The researches on compressing network aim to get efficient training and representation. Recent works on compressing and accelerating CNNs can be classified into four categories: (1) parameter pruning and sharing; (2) low-rank factorization; (3) transferred/compact convolution filters; (4) knowledge distillation [30].

The parameter pruning and sharing based methods explore the redundancy of CNNs and try to remove the redundant parameters. Through investigating vector quantization using information theoretical learning, Gong *et al.* [31] quantized the weights of dense connected layers. They achieved 16-24× compression of the network within 1% loss of classification accuracy on 1000-class classification task in the ImageNet challenge. Similarly, Wu *et al.* [16] took one step further and proposed to quantize both the filter kernels in convolution layers and the weights in fully-connected layers to compress the network storage and computation costs. Chen *et al.* [32] used a low-cost hash function to randomly group connection weights into hash buckets, and through weight-sharing in the same hash bucket. In this way, they compressed the number of weights by a factor 8×. Network pruning have been validated to be a good solution both to reduce network complexity and to prohibit over-fitting.

The low-rank based approaches use matrix/tensor decomposition to estimate the informative parameters of CNNs. By finding an appropriate low-rank approximation of parameters, Denton *et al.* [33] exploited the linear structure of CNN and kept the accuracy degradation within 1% compared to the original model. Tai et al. proposed a new algorithm for computing the low-rank tensor decomposition [34]. Meanwhile, a new method for training low-rank constrained CNNs from scratch was proposed. It was shown to be an effective way to deal with the exploding or vanishing gradients.

The transferred/compact convolution filters based technologies is to design alternative parameter-saving convolution filters or to compact the parameters of CNNs to reduce both the storage usage and computation complexity. Due to the destructive property of binary quantization, the performance of highly quantized network has been validated to be very poor [35]. BinaryConnect [36] used only two possible values (e.g., -1 or 1) to construct CNNs, and updated the parameters using the back propagated error. BinaryConnect achieved state-of-the-art results on small datasets (e.g., CIFAR-10, SVHN), but performed not very well on large-scale datasets [37]. Based on [36], BinaryNet [38] went one step further. It trained CNNs with both binary weights and activations and then replaced most multiplications with 1-bit XNOR operations to reduce memory usage. Different from [36] and [38], XOR-Net [37] proposed to binarize both the filters and the input to convolutional layers, and outperformed BinaryNet by a large margin on ImageNet tasks.

The knowledge distillation methods learn a distilled model and train a more compact CNNs but they reproduce the performance of larger networks. Hinton *et al.* [39] introduced a KD compression framework to ease the training of CNNs through a student-teacher paradigm, in which the student was penalized according to a softened version of the teacher's output. Motivated by LBP descriptor, Juefei-Xu *et al.* [17] proposed an approximation for typical convolutional layer, where the module is comprised of a set of sparse, pre-defined, non-learnable and binary convolution filters together with a 1×1 learnable convolutional layer. Local Binary Convolutional Neural Network (LBCNN) [17] is a related work to our approach, but it has the different orientation encoding mechanism. We also compare with LBCNN on several datasets, and our method outperforms LBCNN by a large margin (as shown in Section IV-C, IV-D). Iandola *et al.* [40] proposed a small CNN architecture named SqueezeNet which achieved on-par performance with AlexNet [1] while showing 50× fewer parameters.

III. ROTATION INVARIANT LOCAL BINARY CONVOLUTIONAL TOPOLOGY

Rotation Invariant Local Binary Convolutional Network (RI-LBCNN) is a CNN with Local Binary orientation Modules (LBoMs). An LBoM is composed of two components, i.e., three layers steerable module (two layers for the first component and one for the second one), which takes advantages of both Local Binary Convolutional [17] and Active Rotating Filters [23]. Specifically, we do not concatenate these two parts explicitly. Instead, we propose to compress network but also show rotation-invariant representations. With LBoMs, RI-LBCNN involves fewer learnable weights, and it is an easily-trained and enhanced deep models.

In the following subsections, we will address four issues in adopting LBoMs in CNNs. Firstly, we give a brief review of LBCNN [17] and ARFs [23]; Secondly, we describe the proposed LBoM and give a detailed analysis of LBoM; Thirdly, we show how LBoM is learned during the back-propagation stage; Finally, a detailed introduction of the RI-LBCNN framework is provided.

A. LOCAL BINARY CONVOLUTIONAL NEURAL NETWORK

Local Binary Convolutional Neural Network is a parameter-saving architecture, which is built by replacing a traditional convolutional layer with an Local Binary Convolutional. LBConv is a three-layer alternative representation for typical convolutional layer. The first layer in LBConv is a set of sparse, fixed and pre-defined binary convolutional layer, followed by an non-linear activation function layer. The last layer is a set of learnable 1×1 convolution weights.

The steps of initializing the LBConv are as follows: First, determine a sparsity level which indicates the percentage of non-zero value weights of the binary convolutional layer. The sparsity is defined as:

$$sparsity = \frac{\#number\ of\ non-zero\ weights}{\#sum\ of\ weights} \tag{1}$$

Secondly, initialize the first convolutional layer through Bernoulli distribution with 0, 1, and -1 randomly based on the *sparsity*. Specially, each location in local binary filters shares an equal probability. Assume that LBConv has p pre-defined non-learnable binary filters, q learnable 1×1 convolution filters. The input image is filtered by p binary filters and becomes p difference maps, which are changed to p bit maps through a non-linear activation layer. Finally, the p bit maps are linearly combined using the q learnable 1×1 weights to approximate the traditional convolutional layer. The weights in first binary convolutional layer are fixed and do not update at the back propagation stage. The learnable weights in 1×1 convolutional layer are updated just like what's in typical convolutional layer.

Because of the sparsity in weights and the fixed binary value of the first convolutional layer, it is less representative compared with the typical convolutional layer. To achieve the similar performance with a traditional convolutional layer at each LBConv, it needs a competitive number of local binary filters (512 in [17]) and 1×1 learnable weight. Compared with typical CNN architecture, an LBConv based network can get on-par performance with fewer learnable parameters, but lacks of the ability to handle with the rotation variations.

B. ACTIVELY ROTATING FILTERS

Actively Rotating Filters like a filter bank in which only one filter being materialized and learned. Let us assume N to be the number of the orientation channels in ARFs. Through clockwise rotating the filters $N-1$ times by $\frac{2\pi n}{N}$, $n = 1, \ldots, N-1$, the remaining $N-1$ rotated filters are obtained without any extra parameters. Compared with typical convolutional layer having the same number of filters, ARFs based convolutional layers (ORConv) can reduce the number of parameters $N-1$ times at a single layer. Feature maps generated by ARFs are not rotation-invariant as orientation information are encoded instead of being discarded. In order to obtain within-class rotation-invariant representation, an alignment/pooling strategy is introduced in [23]. Assume the size of feature map after the last ORConv is $1 \times 1 \times N$. ORAlign is done by first calculating the dominant orientation as D, and then spinning the feature by $-D\frac{2\pi}{N}$. The size of ORAlign output is still $1 \times 1 \times N$. Another rotation invariance encoding strategy is called ORPooling, which is implemented by simply pooling the maximum value among N orientations. This strategy shrinks the output to size of $1 \times 1 \times 1$.

Although ARFs can reduce parameters of CNNs by actively rotating filters, the number of learnable parameters remains large, especially for those popular network architectures like VGGNet [4]. This motivates us to develop filters with binary values.

C. LOCAL BINARY ORIENTATION MODULE

In order to get the maximum approximation to the typical convolutional layer with fewer learnable parameters, as well as to learn the orientation information from raw data, we propose a new Local Binary orientation Module as showed in Fig. 2. Along with orientation-invariant encoding layer, we can get rotation-invariant feature representation. Informally, the LBoM will learn the sparse representation and the orientation information from the input image data simultaneously.

Compared with the conventional convolutional layer, LBoM is a two-part (i.e. three layers) differential module, which consists of a pre-defined non-learnable convolutional layer, an activation function layer and a 1×1 learnable convolutional layer. Different from LBCNN, each convolutional layer in LBoM possesses an additional orientation channel which is obtained through ARFs. Compared with ORNs [23], the parameters of first convolutional layer of LBoM are initialized with 0, 1 and -1, which are fixed and do not updated at the back propagation stage. Thus all of the learnable parameters of LBoM are those learnable parameters in 1×1 convolutional layer (see Fig. 2).

As illustrated in Fig. 2, LBoM starts with p pre-defined non-learnable binary filters (b_i) with N orientation channels, where $i = 1, \ldots, p$. Only the canonical filter (b) is materialized using the Bernoulli distribution with sparsity level s, and the rest $N-1$ orientation channel filters are built using

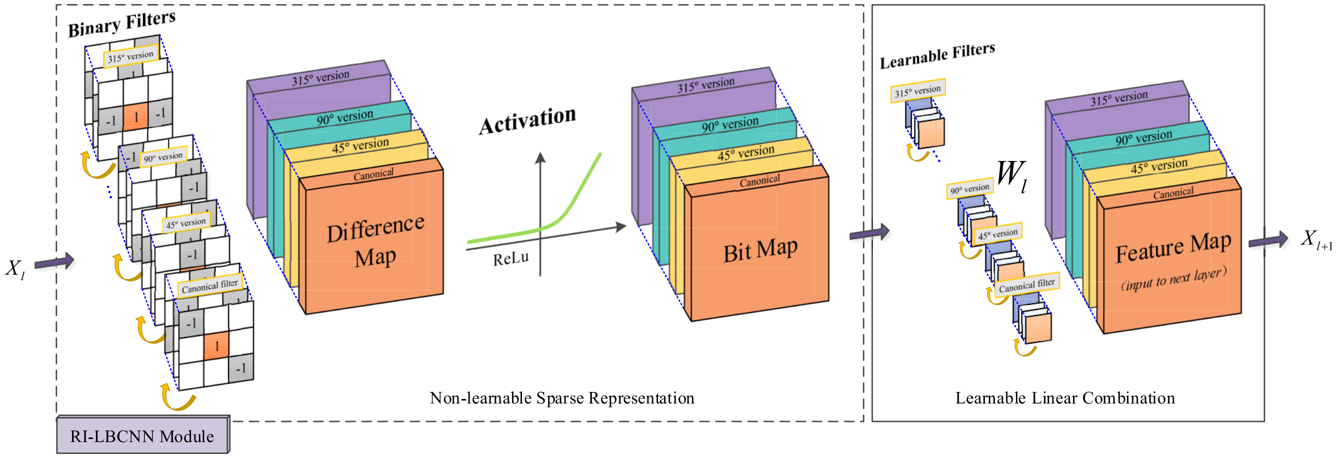

FIGURE 2. Illustration of the Local Binary orientation Module (LBoM) in RI-LBCNNs. This module is an LBoM with 3×3 kernels. Each LBoM consists of two components (three layers), which are the non-learnable sparse representation (box with dash line) and the learnable linear combination (box with solid line). The first convolutional layer is initialized using Bernoulli distribution with 0, 1 and −1. X_l and X_{l+1} are the input and output of the module, respectively. W_l is the learnable weights for linear combination. The extra orientation channel is obtained through clockwise rotating the kernels during convolution (yellow semicircular arrows).

ARFs by clockwise rotating $\frac{2\pi n}{N}$, $n = 1, \ldots, N-1$. The input X_l is filtered by these local binary orientation filters and achieves p **difference maps**, which are then passed into a non-linear activation gate (ReLU in Fig. 2) and the corresponding **bit maps** are produced. These bit maps are only discrete description because of the binary value and the sparsity of the convolution kernels. Afterwards, the p bit maps are linearly combined, which leads to the final output, i.e., **feature maps** of the module. In order to get the corresponding q channels output, here we use 1×1 convolutional layer to convolve with N orientation channels. Among these q learnable 1×1 weights, only q/N weights are learnable, where the rest weights are its rotated copies by ARFs. Finally, the output feature maps X_{l+1} have N orientation channels and the k-th channel is computed as:

$$X_{l+1}^k = \sum_{n=0}^{N-1} F_{\theta_k}^{(n)} * X_l^{(n)}, \theta_k = k\frac{2\pi}{N}, k = 0, \ldots, N-1, \tag{2}$$

$$F_{\theta_k} = \sum_{i=1}^{p} \sigma \cdot b_i^{\theta_k} \cdot W_l^{\theta_k}. \tag{3}$$

Here X_{l+1}^k is the output of (l)-th layer showing k-th orientation; F is the ARF representation of LBoM. F_{θ_k} is the clockwise θ_k-rotated version of F. $F_{\theta_k}^{(n)}$ and $X_l^{(n)}$ are the n-th orientation channle of F_{θ_k} and X_l, respectively. σ is the non-linear binarization operator, which is employed by the ReLU activation. $b_i^{\theta_k}$ is the i-th θ_k-rotated version of binary weights. $W_l^{\theta_k}$ is θ_k-rotated version of learnable 1×1 weights. For the sake of simplicity, there's no bias item in the convolutional layer of LBoM.

Both the first layer of LBoM and LBConv are pre-defined, fixed and do not update at the back propagation stage. But only the canonical filters of the first convolutional layer in LBoM are initialized using the strategy of LBConv, the rest $N-1$ orientation channels are produced by ARFs. Besides, the 1×1 learnable linear weights are initialized using ARFs, which are partial-learnable and are different from those in LBConv. For LBCNN, the number of learnable weights is $p \times q$. For RI-LBCNN, the learnable weights are only part of the 1×1 convolution step, so the number is $\frac{p \times q}{N}$. We then have:

$$\frac{\#params\ in\ LBCNN}{\#params\ in\ RI-LBCNN} = \frac{p \times q}{(p \times q)/N} = N. \tag{4}$$

To achieved within-class rotation invariance, we use ORPooling/ORAlign operator introduced in ORN [23] at the top layer of RI-LBCNN. After adding the rotation encoding layer, the feature representation loses feature arrangement information and RI-LBCNN is rotation invariant.

Because of the binary value and the sparsity of the first convolutional layer in LBoM, the representation capacity of each LBoM is decreased compared to the covolutional layer in the float value. In order to get the on par results, LBoM based CNNs need to enlarge the width of first sparse convolutional layer or the depth of whole network architecture to achieve a better performance.

D. LBOM UPDATING

The training of LBoM is quite straightforward. Gradients are propagated through the fixed convolution kernels just like they do with learnable weights. Note that during the back-propagation stage, we do not update the weights for the binary filters as shown in Fig. 2. Only the weights in the learned filters in 1×1 ORConv layer $W_{l,i}^{\theta_k}$ needs to be updated. And we have:

$$\delta^{(k)} = \frac{\partial L}{\partial F_{\theta_k}}, \theta_k = k\frac{2\pi}{N}, k = 0, \ldots, N-1, \tag{5}$$

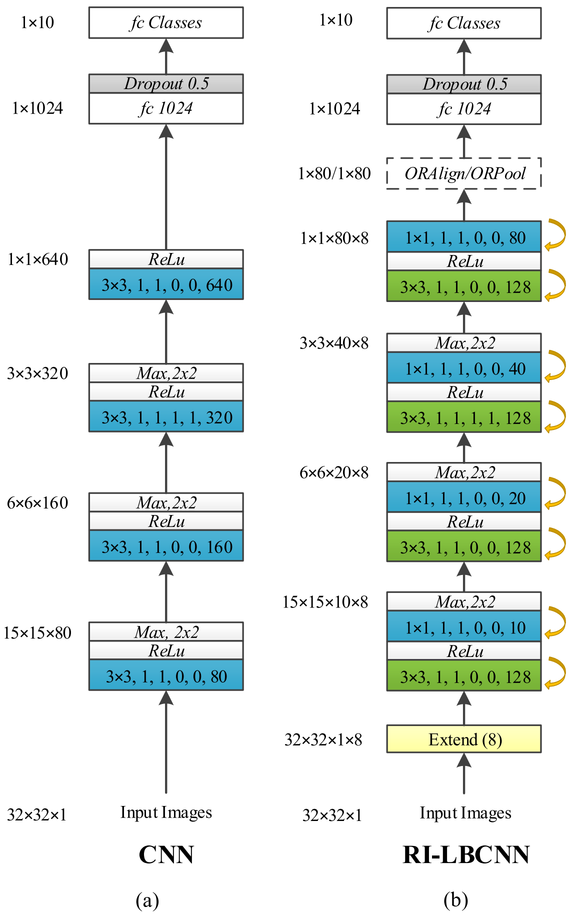

FIGURE 3. Comparison of CNN and RI-LBCNN network architectures. (a) CNN baseline. (b) our RI-LBCNN topology. The left side of each architecture are the size of its output feature maps (e.g., 32×32×1), and the right side is the detailed model.

$$W = W - \eta \frac{\sum_{n=0}^{N-1} \delta_{-\theta_k}^{(k)}}{\sum_{i=1}^{m} \sigma b_i}. \tag{6}$$

where L is the loss function and η is learning rate. Error signals $\delta^{(k)}$ of all the rotated versions of the ARF are assigned to $\delta_{-\theta_k}^{(k)}$. The backward procedure is almost the same as that in ORNs [23], but is different in calculation. From the above description, it can be seen that back propagation process in LBoM is easily to be implemented. By only updating the weights W in learnable 1×1 convolutional layer, the RI-LBCNNs model can be more compact and efficient.

E. ROTATION INVARIANT LOCAL BINARY CONVOLUTIONAL NEURAL NETWORK

In this subsection, we will introduce the architecture of Rotation Invariant Local Binary Convolutional Neural Network. Take the network used in MNIST series as example (Fig. 3(b)), the numbers in the left side are the size of the output at each stage (e.g., 10×1), and the right side are the details of RI-LBCNN network. Here the number of orientation channels is 8. Yellow box is the extend layer for data augmentation; Green boxes are non-learnable sparse convolutional layer with pre-defined binary weights; Blue boxes are learnable ORConv layer. Both of these two layers have an additional orientation channel, which are obtained through actively rotating the convolution filters clockwise (yellow semicircular arrows); Box with dash line is optional for the network; Gray boxes are dropout layer. Meanwhile, the numbers in the layer boxes are the hyper parameters for each layer. Taking the parameters of the first convolutional layer as example (3×3, 1, 1, 0, 0, 128), it means the convolutional layer consists of 128 kernels, and the kernel size is 3×3. Meanwhile, the *stride* for both x and y direction are 1, and the *padding* for both x and y direction are 0. Giving an input image of size 32×32×1, the extend layer is used to duplicate the input image 8 times. The augmented data is then fed to an LBoM. Different from the traditional convolutional layer, the ReLU layer in LBoM is used only after the binary convolutional layer instead of each convolutional layer, and the ORConv layer is employed to learn the combination of the sparse binary representation. To reduce the computation cost, max-pooling layer is employed here to downsize the feature maps. After a bunch of LBoMs, the orientation information is encoded in the feature maps instead of being discarded in the traditional CNN. Thus the ORPooling/ORAlign layer is employed to obtain the rotation-invariant representation. Finally, the network outputs the class label of the input image.

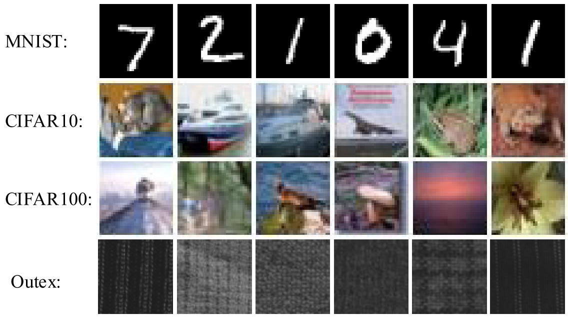

FIGURE 4. The sample images of the four datasets.

IV. EXPERIMENTS

In this section, we present the experimental results on four benchmarks, and the sample images for each dataset are illustrated in Fig. 4. The first one is MNIST and its variants, including MNIST [41], MNIST-rot-12k [42] (a small rotation version of MNIST), and MNIST-rot used in ORN [23]. Firstly, the ablation experiment is conducted to explore the details of LBoM. Secondly, we use both the MNIST and MNIST-rot datasets to show the advantages of RI-LBCNN on its rotation invariance. Furthermore, RI-LBCNN is tested on MNIST-rot-12k to validate its generalization ability on rotation invariance. Thirdly, we validate our proposed networks are tested on texture classification. Finally, we upgrade the VGGNet [4], ResNet [6] and WideResNet [5] network architectures using LBoMs, and applied the upgraded architectures on CIFAR-10 and CIFAR-100 [43], i.e., two natural image classification datasets, to further evaluate the performance of our LBoMs on image classification.

A. EXPERIMENTAL SETTINGS

For all the three MNIST datasets, we use the same network topology. The baseline CNN we adopt is as introduced in [23], which is composed of four convolutional layers with multiple 3×3 kernels, as illustrated in Fig. 3(a). We build another LBCNN baseline through replacing each convolutional layer in CNN with LBConv [17]. In addition, we reimplement ORNs through replacing typical convolutional layer with ARFs following the idea of [23]. The comparison of CNN [23] and RI-LBCNN is illustrated in Fig. 3. RI-LBCNN is generated by upgrading convolutional layers in baseline CNN using LBoM with 4 or 8 orientation channels. To obtain the rotation-invariant representation, ORAlign/ORPooling layer introduced in [23] is employed at the top layer of RI-LBCNN architectures. In order to make a fair comparison, we make a trade-off between representative and learnable weights saving. Considering the LBoM module possesses an additional orientation channel, we decrease the number of kernels of the first sparse binary layer to one quarter (128 vs. 512 in LBCNN), and the second learnable convolutional layer to one-eighth. Therefore, the complexity of RI-LBCNN is reduced compared with CNN (see the fourth column of Tab. 2).

In experiments, we use the same hyper parameters as that in [23], and we perform the training using tuning-free convergent Adadelta algorithm [44], with the training epochs (50), batch size (128) and dropout rate (0.5) for the fully-connected layers. Our implementation is based on Torch [45]. We run experiments on CPU i7, 128GB RAM and GeForce GTX TITAN X (12G).

B. MNIST DATASETS

1) MNIST

The original *MNIST* [41] dataset is a very typical dataset to verify the performance of the proposed method. MNIST contains a training set of 60k and a testing set of 10k with 32×32 gray scale images showing the handwritten digits from 0 to 9.

2) MNIST-rot-12k

MNIST-rot-12k [42] is a commonly used dataset for validating rotation-invariant algorithm. It consists of images from a subset of original MNIST, rotated by a random angle in $[0, 2\pi]$. This dataset contains 12k training samples and 50k testing samples. Among them, 2k images are randomly selected as validation set and the remaining 10k images are training set.

3) MNIST-rot

To verify the rotation invariance of the proposed RI-LBCNN, the *MNIST-rot* dataset introduced in [23] is used. The reasons that we choose the MNIST-rot dataset are two-fold. Firstly, the scale of *MNIST-rot-12k* is smaller than that of the *MNIST-rot* (12k vs. 60k); Secondly, there are some limitations for the training images (fewer rotation variations for examples) in *MNIST-rot-12k*. So Zhou *et al.* [23] proposed to take the full MNIST dataset, generated the training set with a random angle in the range of $[0, 2\pi]$. This simple strategy not only enlarges the size of the training data, but also increases the diversity of rotation variations.

TABLE 1. Error rates on MNIST-rot compared with networks based on A-LBoM.

Method	LBoM	A-LBoM
RI-LB-4-ORPool	**31.89**	56.32
RI-LB-4-ORAlign	**30.72**	56.74
RI-LB-8-ORPool	**25.6**	57.3
RI-LB-8-ORAlign	**23.47**	56.09

C. ABLATION EXPERIMENTS

In this subsection, we compare RI-LBCNN with networks that based Ablated Local Binary orientation Module (A-LBoM). A-LBoM is obtained through removing ARFs in the second part, i.e., the 1×1 convolutional layer in LBoM. Results in Tab. 1 show that networks based on LBoM can learn the additional orientation information and preform better. Networks based on A-LBoM work poor compared to those networks based on LBoM. One reason is that the additional orientation channel obtained from LBoM will encode the rotation information of an input image separately. Through the ORAlign/ORPooling layer, the separated orientation information will obtain rotation-invariant representation without losing the discriminative capacities. Networks based on A-LBoM will result in a confusion of object representation and object orientation representation, which will result in an obvious decrease of the network performance (23.47% vs. 56.09%). Thus, a separate orientation channel all along with the module is recommended when LBoM is employed in the network for the rotation invariance.

D. EXPERIMENT ANALYSIS ON MNIST

1) MNIST AND MNIST-rot

For both MNIST and MNIST-rot, we randomly select 10k images for validation, and the rest 50k images for training. The best model is selected by 5-fold cross validation and then is applied to the test set. The results on MNIST and MNIST-rot are presented in Tab. 2. Besides, the state-of-the-art STN [21], TI-Pooling [22] and ORNs [23] are involved in the comparison.

To evaluate the influence of the hyper-parameter *sparsity*, by ranging the values from 0.1 to 0.9, we conduct experiments on MNIST and MNIST-rot using RI-LBCNN architecture. As is shown in Fig. 5, the influence of the sparsity level on the classification accuracy is marginal when the *sparsity* is changed, the best classification accuracy (all these three types are considered) is achieved when $sparsity = 0.4$. While for the fair comparison with LBCNN [17], we take the fixed sparsity level $sparsity = 0.5$ at all the following experiments.

In Tab. 2, the second column refers to the weight type of convolutional layers; The third column refers to the number of convolutional kernels in each layer, and a similar

TABLE 2. Classification error rates on the MNIST and MNIST-rot datasets.

Methods	Weight Type	#network conv kernels	#params(M)	original(%)	rot(%)	original → rot(%)
LBCNN-baseline [17]	binary+float	80-160-320-640	1.18	0.8	3.46	58.68
CNN-baseline	float	80-160-320-640	3.08	0.73	2.82	56.28
STN [21]		80-160-320-640	3.20	0.66	2.88	55.59
TI-Pooling [22]		(80-160-320-640)$_{\times 8}$	24.64	0.97	-	-
ORN-8(ORAlign) [23]		10-20-40-80	0.96	**0.59**	1.42	**16.21**
Ours						
RI-LBCNN-8(ORPooling)	binary+float	10-20-40-80	**0.39**	0.97	1.36	25.77
RI-LBCNN-8(ORAlign)		10-20-40-80	0.96	0.73	**1.25**	23.46

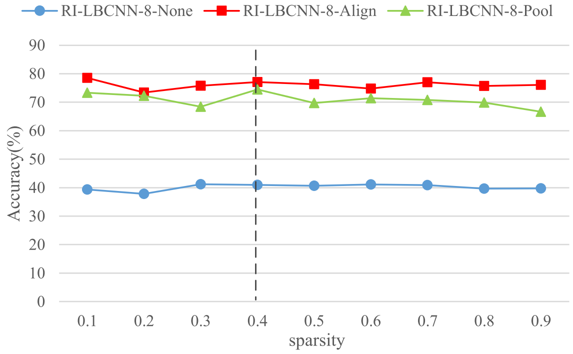

FIGURE 5. The classification performance comparison on the MNIST dataset of RI-LBCNN using different sparsity during initialization.

TABLE 3. Classification error rates on the MNIST-rot-12k datasets.

Method	Weight Type	Error(%)
ScatNet-2 [46]	float	7.48
PCANet-2 [47]		7.37
CNN		4.34
LBCNN [17]	binary+float	6.39
RI-LBCNN-8(ORAlign)		**3.05**

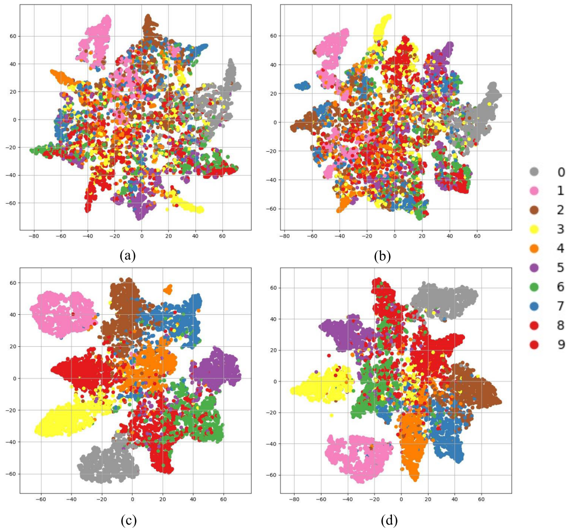

FIGURE 6. Visualization of the features on MNIST-rot dataset, corresponding to the last column of Tab. 2. (a) CNN. (b) LBCNN. (c) RI-LBCNN(ORAlign). (d) RI-LBCNN(ORPool).

notation is also used in [5]. In each LBoM (4 in total), we use the same hyper-parameter as that in LBCNN [17]. We use 0.5 for sparsity, and 128 local binary filters in the first sparse layer. The performance comparison is shown in the last three columns in terms of error rate. By comparing with baseline CNN, RI-LBCNN achieves better performance with LBoM module but only using 1/3, 1/6 parameters of CNN. On the original MNIST dataset without rotation, RI-LBCNN with 8 orientations and ORAlign operator achieves 0.73% test error.

Moreover, in MNIST-rot datasets, the performance on baseline CNN and LBCNN model degrade due to rotation, while ORN and RI-LBCNN can capture orientation features and achieve better results. Meanwhile, our RI-LBCNN-8 (ORAlign) outperforms ORN-8 (ORAlign) on the rotated MNIST samples. However, when deploy models trained on original MNIST and MNIST-rot (the last column in the Tab. 2), our RI-LBCNN (23.46%) performs not so well compared with ORN (16.21%). The reason causes this differences is that with the parameter-saving strategy in the first binary convolutional layer, the sparse binary-value feature response in LBoM is less robust than that of float-value based ORN. Finally, RI-LBCNN gains 60% improvement on the classification performance over LBCNN baseline. Based on the t-SNE [14] feature visualization techniques, it also can be seen in Fig. 6 that the features produced by RI-LBCNN are more discriminative than those of by CNN and LBCNN [17].

2) MNIST-rot-12k

Because the sample resolution in MNIST-rot-12k is smaller than that of MNIST (32×32 vs. 28×28), we make a minor modification of the network architecture to fit the input.

The kernel padding of the second convolutional layer is 1, compared to 0 for MNIST. We train this network on a single GPU for 200 epochs and compare the performance with the state-of-the-art results published on this dataset. We test the RI-LBCNN-8 that uses LBoM with 8 orientation channels and an ORAlign layer to encode the orientation information. Tab. 3 shows that RI-LBCNN-8 (ORAlign) can decrease the error rate from 6.39% to 3.05% using only 75% network parameters of the baseline LBCNN.

E. TEXTURE CLASSIFICATION

We test the proposed RI-LBCNN in Outex datasets to verify the performance on texture classification since textures always show rotation variations. *Outex-TC-00011-r* consists of a 24-class gray-scale textures, which contains 20 samples per class with the size of 120×120 or 100×100 each. All the samples are obtained under an illumination of *inca*. To reduce

TABLE 4. Classification error rates on the Outex datasets.

Method	Weight Type	ori → rot(%)
CNN	float	43.57
ORN-8(ORAlign) [23]		48.79
LBCNN [17]	binary	47.03
RI-LBCNN-8(ORAlign)		**33.34**

the over-fitting on this dataset because the limited number of images (480 samples totally), we enlarge the Outex dataset manually using the label-preserving transformations. We do this by random extracting 32×32 patches from the source image with an augmentation factor of 20, and training the network on these extracted patches. Finally, we built a dataset which contains 9000/600/9600 samples for train/val/test separately. The rotation samples are generated through rotating image samples with a random angle in the range of [0, 360]. From Tab. 4, we can see that the proposed RI-LBCNN achieved the best performance by a large margin(33.34% vs. 48.79%) when the model is trained on original data and deploy on the rotation samples. RI-LBCNN shows strong generalization capacity on the classification of the rotated patterns, though based on partly-fixed binary parameters. The reason for the result is that: Compared with non-directional binary filters learned by LBCNN, the oriented response filters in RI-LBCNN can learn highly directional patterns, and some of them are quite suitable for the recognition of the texture samples. Meanwhile, both the LBCNN and RI-LBCNN are benefitted from the data augmentation techniques. We also find that the binary-value network RI-LBCNN outperforms those float-value CNNs, which learn the texture patterns by rote. The experiments on Outex validate the robustness of RI-LBCNN to the rotation variations.

F. NATURAL IMAGE CLASSIFICATION

For the natural image classification task, we use the CIFAR10 and CIFAR100 datasets [43]. CIFAR dataset contains 60k samples, which is composed of a training set of 50K and a testing set of 10K. Images in CIFAR dataset are 32×32 color images and have 10 or 100 classes respectively. In each class, it respectively contains 6000 or 600 images.

CIFAR datasets contain a variety of categories with object local/global orientation variations. First, we upgrade three state-of-the-art CNNs including VGGNet [4], ResNet [6] and WideResNet [5]. We use the LBoMs to replace the typical convolutional layers in these CNNs, and the bottlenecks in ResNet and wideResNet are replaced with 1×1 LBoM. The convolutional layers in the main branch are replaced with LBoMs, which consists of 3×3 LBoMs, BatchNorm, ReLU and followed by 1×1 ORConv layer to learn the linear combination of the binary sparse representation.

From Tab. 5, it can be seen that CNNs using LBoMs outperform those using LBConv on all three architectures but with fewer parameters. RI-LB-VGG uses only 0.05% learnable parameters compared to VGG itself, and it achieves 16.1% (vs 20.75% for VGG) and 46.98% (vs 59.02% for VGG) error rate on CIFAR-10 and CIFAR-100, respectively. With regard to ResNet, RI-LB-ResNet uses only one-quarter learnable parameters of LB-ResNet, but achieves an accuracy improvement both on CIFAR-10 and CIFAR-100 significantly. These three architectures are all composed of wider kernels and considerable network depth, which means the redundancy of the network parameters. Compared with LBConv based networks, the additional orientation channels in RI-LBCNN will result in a small performance improvement. Specifically, the newly-added orientation channels contribute to the performance gain on these rotated samples, Fig. 7.

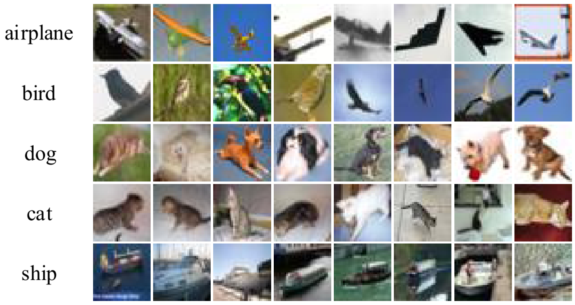

FIGURE 7. CIFAR10 sample images that contain rotated objects falsely classified by the LB-VGGNet but correctly recognized by the proposed RI-LB-VGGNet.

V. DISCUSSION

In this paper, we propose an Local Binary orientation Module, with which an Rotation Invariant Local Binary Convolutional Neural Network is proposed for both the robustness to the rotation invariance and network compression. In this section, we now discuss some computation complexity and rotation invariance afforded by LBoM over a typical convolutional layer.

A. COMPUTATION COMPLEXITY

RI-LBCNN reduce the number of learnable parameters by a factor 1.24× to 1675× compared with traditional CNNs, and 3× to 250× compared with VGG, ResNet and wide-ResNet upgraded using LBConv. What's more, the binary nature of the first layer of LBoM and the active rotating mechanism on convolution filters can further reduce the number of learnable parameters and computation cost both during training and inference. Experiments in Section IV show that the employed rotation-encoding mechanism achieves more performance gains on the classification accuracy compared with LBConv based models. Additionally, RI-LBCNN with fewer learnable parameters can reduce the risk of over-fitting. Conventional CNNs with high performance usually use Dropout [48] and BatchNorm [27] to prevent over-fitting and reduce internal co-variant shift.

B. ROTATION INVARIANCE

Compared with LBCNN and CNN, our RI-LBCNN can handle rotation variance better, i.e., handling unseen rotated

TABLE 5. Classification error rates on CIFAR-10 and CIFAR-100 image classification dataset, respectively. *k* is the widening factor introduced in WideResNet [5].

Method	Weight Type	#depth-k	#params	CIFAR-10(%)	CIFAR-100(%)
LB-VGG	binary+float	16	3.0M	20.65	59.02
RI-LB-VGG		16	0.012M	**16.1**	**46.98**
LB-ResNet	binary+float	20	4.2M	29.97	51.64
RI-LB-ResNet		20	1.1M	**21.98**	**50.77**
LB-WideResNet	binary+float	16-4	0.47M	16.07	47.83
		16-8	0.93M	15.66	44.07
RI-LB-WideResNet		16-4	0.23M	**15.23**	**46.39**
		16-8	0.47M	**14.46**	**43.49**

samples based on LBoM (see Tab. 2 and 5). The results on MNIST and MNIST-rot have shown that RI-LBCNN outperforms both CNN and LBCNN.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduce an Local Binary orientation Module (LBoM). It contains a set of pre-defined binary, active rotating, and randomly generated convolution filters and a set of rotated learnable filters. Based on LBoM, a novel network architecture named Rotation Invariant Local Binary Convolutional Neural Network (RI-LBCNN) is proposed, which is parameter-saving and orientation-invariant. RI-LBCNN exploits the domain to construct neural networks with fewer learnable parameters while endows features showing the capacity of orientation invariance. Meanwhile, it improves CNNs on the generalization ability of rotation by introducing an extra orientation channel. Extensive experiments show that RI-LBCNNs can achieve on-par or better performance with the state-of-the-art performance over several benchmarks but with fewer parameters. In the future, RI-LBCNN can be further improved in the following directions: 1) New strategies to boost the discriminability of RI-LBCNN; 2) Reduction of the computational cost; 3) Application of RI-LBCNN to large-scale computer vision tasks including object detection and image retrieval.

ACKNOWLEDGMENT

Tekes, Academy of Finland and Infotech Oulu are also gratefully acknowledged.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[3] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[4] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[5] S. Zagoruyko and N. Komodakis. (2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146

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

[7] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[9] R. Girshick, "Fast R-CNN," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[11] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, pp. 1–4.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2999–3007.

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

[14] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[15] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, Apr. 2016.

[16] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4820–4828.

[17] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1–10.

[18] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[19] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[20] K. Liu *et al.*, "Rotation-invariant HOG descriptors using Fourier analysis in polar and spherical coordinates," *Int. J. Comput. Vis.*, vol. 106, no. 3, pp. 342–364, 2014.

[21] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[22] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys, "TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 289–297.

[23] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4961–4970.

[24] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. (2016). "Rotation equivariant vector field networks." [Online]. Available: https://arxiv.org/abs/1612.09346

[25] X. Zhang, L. Liu, Y. Xie, J. Chen, L. Wu, and M. Pietikäinen, "Rotation invariant local binary convolution neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, Oct. 2017, pp. 1210–1219.

[26] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *J. Comput. Graph. Stat.*, vol. 10, no. 1, pp. 1–50, 2001.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[28] X. Weng, S. Wu, F. Beainy, and K. Kitani. (2017). "Rotational rectification network for robust pedestrian detection." [Online]. Available: https://arxiv.org/abs/1706.08917

[29] M. Denil, B. Shakibi, L. Dinh, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[30] C. Yu, W. Duo, Z. Pan, and Z. Tao. (2017). "Model compression and acceleration for deep neural networks." [Online]. Available: https://arxiv.org/abs/1710.09282

[31] Y. Gong, L. Liu, M. Yang, and L. Bourdev. (2014). "Compressing deep convolutional networks using vector quantization." [Online]. Available: https://arxiv.org/abs/1412.6115

[32] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.

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

[34] C. Tai, T. Xiao, Y. Zhang, and X. Wang. (2015). "Convolutional neural networks with low-rank regularization." [Online]. Available: https://arxiv.org/abs/1511.06067

[35] M. Courbariaux, Y. Bengio, and J.-P. David. (2014). "Training deep neural networks with low precision multiplications." [Online]. Available: https://arxiv.org/abs/1412.7024

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

[37] 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.

[38] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. (2016). "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1." [Online]. Available: https://arxiv.org/abs/1602.02830

[39] G. Hinton, O. Vinyals, and J. Dean. (2015). "Distilling the knowledge in a neural network." [Online]. Available: https://arxiv.org/abs/1503.02531

[40] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[41] 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.

[42] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 473–480.

[43] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[44] M. D. Zeiler. (2012). "ADADELTA: An adaptive learning rate method." [Online]. Available: https://arxiv.org/abs/1212.5701

[45] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *Proc. BigLearn NIPS Workshop*, 2011, pp. 1–6.

[46] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[47] T.-H. Chan, K. Jia, S. Gao, J. Lu, and Z. Zeng, Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *IEEE Trans. Image Process*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

XIN ZHANG received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 2011 and 2013 respectively, where he is currently pursuing the Ph.D. degree in control science and engineering. His research interests include image/video processing and deep learning.

YUXIANG XIE received the B.S., M.S., and Ph.D. degrees in systems engineering from the National University of Defense Technology in 1998, 2001, and 2004, respectively. She is currently an Associate Professor with the School of Information System and Management, National University of Defense Technology. Her research interests include image and video analysis, classification, and retrieval.

JIE CHEN (M'07) received the M.S. and Ph.D. degrees from the Harbin Institute of Technology, China, in 2002 and 2007, respectively. Since 2007, he has been a Senior Researcher with the Center for Machine Vision and Signal Analysis, University of Oulu, Finland. In 2012 and 2015, he visited the Computer Vision Laboratory, University of Maryland at College Park and the School of Electrical and Computer Engineering, Duke University, respectively. His research interests include pattern recognition, computer vision, machine learning, dynamic texture, deep learning, and medical image analysis. He has authored over 60 papers in journals and conferences. He was a Co-Chair of the International Workshops at ACCV, CVPR, and ICCV. He was a Guest Editor of the *Journal of Neurocomputing* and TPAMI. He is an Associate Editor of *The Visual Computer*.

LINGDA WU received the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China. She is currently a Professor with Space Engineering University, Beijing, China. Her research interests include multimedia information systems and virtual reality technology.

QIXIANG YE (SM'15) received the B.S. and M.S. degrees in mechanical and electrical engineering from the Harbin Institute of Technology, China, in 1999 and 2001, respectively, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2006. He was a Visiting Assistant Professor with the Institute of Advanced Computer Studies, University of Maryland at College Park until 2013. He has been a Professor with the University of Chinese Academy of Sciences since 2016. He has authored or co-authored over 80 papers in refereed conferences and journals. His research interests include image processing, visual object detection, and machine learning. He received the Sony Outstanding Paper Award.

LI LIU received the B.S. degree in communication engineering, the M.S. degree in photogrammetry and remote sensing, and the Ph.D. degree in information and communication engineering from the National University of Defense Technology, China, in 2003, 2005, and 2012, respectively. During her Ph.D. study, she spent over two years as a Visiting Student at the University of Waterloo, Canada, from 2008 to 2010. From 2015 to 2016, she visited the Multimedia Laboratory at the Chinese University of Hong Kong. From 2016 to 2018, she is visiting the Center for Machine Vision and Signal analysis at the University of Oulu, Finland. She joined the faculty at the National University of Defense Technology in 2012, where she is currently an Associate Professor with the College of Information System and Management. Her research interests include texture analysis, image classification, object detection, and scene understanding.