Machine Intelligence Research
www.mi-research.net

19(2), April 2022, 115-126
DOI: 10.1007/s11633-022-1324-5

# Sharing Weights in Shallow Layers via Rotation Group Equivariant Convolutions

Zhiqiang Chen[1] Ting-Bing Xu[2] Jinpeng Li[3] Huiguang He[1,4]

[1] Research Center for Brain-inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2] School of Instrumentation and Optoelectronic Engineering, Beihang University, Beijing 100191, China
[3] Ningbo HwaMei Hospital, University of Chinese Academy of Sciences, Ningbo 315012, China
[4] Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Beijing 100190, China

**Abstract:** The convolution operation possesses the characteristic of translation group equivariance. To achieve more group equivariances, rotation group equivariant convolutions (RGEC) are proposed to acquire both translation and rotation group equivariances. However, previous work paid more attention to the number of parameters and usually ignored other resource costs. In this paper, we construct our networks without introducing extra resource costs. Specifically, a convolution kernel is rotated to different orientations for feature extractions of multiple channels. Meanwhile, much fewer kernels than previous works are used to ensure that the output channel does not increase. To further enhance the orthogonality of kernels in different orientations, we construct the non-maximum-suppression loss on the rotation dimension to suppress the other directions except the most activated one. Considering that the low-level-features benefit more from the rotational symmetry, we only share weights in the shallow layers (SWSL) via RGEC. Extensive experiments on multiple datasets (i.e., ImageNet, CIFAR, and MNIST) demonstrate that SWSL can effectively benefit from the higher-degree weight sharing and improve the performances of various networks, including plain and ResNet architectures. Meanwhile, the convolutional kernels and parameters are much fewer (e.g., 75%, 87.5% fewer) in the shallow layers, and no extra computation costs are introduced.

**Keywords:** Convolutional neural networks (CNNs), group equivariance, higher-degree weight sharing, parameter efficiency.

**Citation:** Z. Chen, T. B. Xu, J. Li, H. He. Sharing weights in shallow layers via rotation group equivariant convolutions. *Machine Intelligence Research*, vol.19, no.2, pp.115–126, 2022. http://doi.org/10.1007/s11633-022-1324-5

## 1 Introduction

Convolutional neural networks (CNNs)[1] have developed rapidly in the past two decades. In various tasks of computer vision, including classification[2], detection[3], and semantic segmentation[4], CNNs have achieved excellent performance. CNNs have translation group equivariance and share weights in different positions. Compared to fully-connected networks[5], CNNs have significantly higher parameter efficiency due to weight sharing, and can better resist the influence of translation, thus achieving better performance. Inspired by this, we aim to achieve better performance with an even higher degree of parameter efficiency[6–8].

In 2016, Cohen and Welling[9] proposed group equivariant convolutional neural networks(G-CNNs). It extended the translation group equivariance of traditional convolutions to more group equivariances (e.g., rotation group equivariance, mirror group equivariance). When changing the sampling grids of a kernel, the shape of the kernel will be irregular. It results that G-CNN can only work on square lattices (i.e., rotation angles of $i\pi/2$) and therefore is limited in rotational symmetries. In order to exploit more symmetries, some methods design special convolutions such as the steerable filter CNNs (SFCNN)[10], general E(2)-equivariant steerable CNNs (E2CNN)[11], and partial differential operator based equivariant convolutions (PDO-eConvs)[12]. They combined some basic kernels (e.g., Gaussian radial profiles, partial differential operators) and conducted rotation by changing the weights of the basic kernels. However, these indirect approaches increase the difficulty of applying them to ordinary networks.

Furthermore, previous approaches paid more attention to the number of parameters, often introducing more computational costs. Most previous works, such as G-CNNs, SFCNN, E2CNN and PDO-eConvs, rotated each convolution kernel to multiple orientations. As a result, they consumed much more computational costs under the same parameters. Especially for SFCNN, E2CNN and PDO-eConvs, they combined some base kernels to generate final kernels in convolution operations. The size of the

Research Article
Manuscript received December 1, 2021; accepted February 7, 2022; published online March 7, 2022
Recommended by Associate Editor Ming-Ming Cheng
Colored figures are available in the online version at https://link.springer.com/journal/11633

base kernel is lager than the conventional kernel (7×7 or 9×9 for SFCNN and E2CNN, 5×5 for PDO-eConvs). Thus, each convolution operation will consume more computational costs than the conventional convolution with the same parameters.

This paper exploits and explores a novel method to take more advantage of the rotation group equivariant convolution (RGEC). In addition to improving the efficiency of the network parameters, we also ensure that no additional computational burden is introduced. To this end, we keep the output channels of the RGEC the same as those in the ordinary convolutions, rather than increasing the output channels to keep the number of parameters alike. Therefore, no computing resources are increased, whereas the kernels and parameters are much fewer. To avoid introducing extra computational cost, we construct an arbitrary shape convolution, which can rotate the convolution kernels directly and conveniently. Considering that the RGEC is more efficient for the low-level features, we construct networks that share weights of different orientations only in shallow layers. With fewer parameters and no extra computational burdens, a non-maximum-suppression loss on the orientation dimension is designed and added to improve performance. Extensive experiments demonstrate that sharing weights for different orientations in the shallow layers can improve the performance with fewer parameters and no extra computations when utilizing a drop-in replacement for conventional convolutions.

## 2 Related work

### 2.1 Convolutional neural networks

Unlike fully-connected neural networks[5], CNN shares weights in different positions, so that it can achieve better performance with fewer parameters in many computer vision tasks. LeNet-5[1] introduced the convolution layer to neural networks, which initiated extensive research on convolution neural networks. After that, more complex and deeper CNNs such as AlexNet[13], VGG[14], GoogleNet[15], ResNet[16] and Densenet[17] are proposed and achieved state of the art in image classification[18], detection[19–24], semantic segmentation[25–27], etc.

### 2.2 Rotation group equivariant convolution

CNN owns the property of the translation group equivariance. To extend the equivariance[28] to a larger group[29], RGEC was proposed[9, 30, 31]. Some works obtained rotation-invariant by rotating input images[32, 33]. These methods cannot benefit from the rotational symmetries[34] of features. G-CNNs[9] used a rotation group convolution to exploit rotation symmetries of features and enjoy a higher degree of weight sharing. Due to the irregular shape of the convolution kernels[35, 36], G-CNNs can only work on square lattices and achieve 4-orientation symmetries. To exploit more symmetries, SFCNN[10] and PDO-eConvs[12] designed special convolution kernels composed of some basic kernels. They rotated the whole kernel by indirectly changing the weights of basic kernels.

## 3 Methodology

CNNs have much higher parameter efficiency than fully connected networks and achieve better performance in various tasks. According to Occam′s Razor[37], fewer parameters will take less generalization risk. To further improve the parameter efficiency, we incorporate the RGEC into the existing neural architectures.

### 3.1 Review of rotation group equivariant convolution

As defined in G-CNN[9], group equivariance satisfied:

$$\Phi(T_g(f)) = T'_g(\Phi(f)) \tag{1}$$

where $T_g$ is a transformation operation with group $g$, $f$ is the input, and $\Phi$ is the feature mapping function. For a rotation group,

$$G = \{g_0, g_1, \cdots, g_{n-1} | g_i = i\frac{2\pi}{n}\}. \tag{2}$$

$T_{g_i}(f), g_i \in G$ means rotating $f$ with degree of $g_i$. To simplify it, we denote $T_{g_1}(f)$ as $r(f)$, and $T_{g_i}(f)$ as $r^i(f)$, which represents the $r(\cdot)$ operation $i$ times. For a general convolution kernel $k$, when we rotate the feature map $f$ with $g_i$, the convolution can be represented as

$$r^i(f) * k = r^i(f) * r^i(r^{-i}(k)) = r^i(f * r^{-i}(k)) \tag{3}$$

where $r^{-i}(\cdot)$ represents rotating the target with degree $-g_i$, and $*$ is convolution operation.

In order to obtain the rotation group equivariance as well as share weights in different orientations, a kernel $k$ can be expanded to group kernels $K = [r^0(k), r^1(k), \cdots, r^{n-1}(k)]$, the convolution process is

$$f * K = [f * r^0(k), f * r^1(k), \cdots, f * r^{n-1}(k)]. \tag{4}$$

When we rotate the feature map $f$ with degree $g_i$, it can be represented as

$$\begin{aligned} r^i(f) * K = r^i(f * r^{-i}(K)) = \\ r^i([f * r^{-i}(k), f * r^{1-i}(k), \cdots, f * r^{n-1-i}(k)]) = \\ r^i(f * shift^i(K)) = \\ r^i(shift^i(f * K)) \end{aligned} \tag{5}$$

where $shift^i(\cdot)$ is shifting the target cyclically by $i$ elements and $shift^1([x_1, x_2, \cdots, x_n]) = [x_n, x_1, \cdots, x_{n-1}]$. If we regard $r^i(\cdot)$ as $T_{g_i}(\cdot)$, and $r^i(shift^i(\cdot))$ as $T'_{g_i}(\cdot)$, the convolution with kernel $K$ satisfies the group equivariance on $G$.

For a subsequent convolution, we shift the feature maps cyclically before the convolution in each orientation. Specifically, the group kernels are denoted as $K_1 = [r^0(k_1), r^1(k_1), \cdots, r^{n-1}(k_1)]$, then the convolution is defined as

$$F_{K_1}(x) = [shift^0(x)*r^0(k_1), \cdots, shift^{-(n-1)}(x)*r^{n-1}(k_1)]. \tag{6}$$

Easy to prove that it still satisfies the rotation group equivariance on $G$ when we stack multiple convolution layers (see detail proof in Appendix):

$$\begin{gathered} F_{K_1}(r^i(f) * K) = r^i(shift^i(F_{K_1}(f * K))) \\ F_{K_2}(F_{K_1}(r^i(f) * K)) = r^i(shift^i(F_{K_2}(F_{K_1}(f * K)))) \\ \cdots \end{gathered} \tag{7}$$

Fig. 1 illustrates the posture relationship between kernels in different layers. $f$ means the input image. $k$ means the original convolutional kernel. $r^i(k)$ means rotating the original kernel. $K$ is the group kernel after a series of rotations. After $f * K$, we can get a series of feature maps. $shift^{-i}$ means changing the order of the feature maps after convolution in the Previous layer. $F_{K_1}$ means the convolution in the second layer, where $K_1$ is a series of kernels rotated from the original kernel $k_1$. One point of the kernel in a deeper layer can be regarded as representing a whole kernel in its previous layer. To illustrate the posture relationship, we replace the left-top point of the kernel in the deeper layer with the whole kernel in the previous layer. As Fig. 1 shows, two stacked convolution layers with rotation group equivariance can hold a stable kernel posture relationship between layers. Among different orientations, networks share the same parameters with the same structure. As a result, the stacked RGEC can maintain rotation group equivariance.

In the above description, only the convolution operation is discussed. In prevalent CNNs, the batch normalization (BN) and rectified linear units (ReLU) layers are always indispensable. It can also be proved easily that the ReLU operation has the characteristic of rotation group equivariance. As for the BN layer, it can be equivariant when it shares the parameters (e.g., mean, variance) in different orientations.

## 3.2 Arbitrary shape convolution

As the convolution kernel is rotated, the shape of the kernel becomes irregular. It has been found in many previous works that a special kernel is designed to avoid resulting in an irregular kernel. However, the base kernels are usually bigger than the original kernels, introducing plenty of extra computations. In our approach, we propose an arbitrary shape convolution, therefore, the kernel can be rotated directly without introducing base kernels.

For a convolution kernel, it can be represented as weight parameters $W(k) = [a_1, a_2, \cdots, a_n]$ and their corresponding location recordings $P(k) = [(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)]$ (see in Figs. 2 (c) and 2 (d)). For example, a general $3 \times 3$ kernel

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \tag{8}$$

can be represented as weight parameters $[a_1, a_2, \cdots, a_9]$ and corresponding location recordings $[(-1, -1), (0, -1), \cdots, (1, 1)]$ (central element $a_5$ locates (0,0)). In this way, the convolution kernel can be arbitrary shape.

Define:

$$Stack_{-P(k)}(f) = [s_{(-x_1, -y_1)}(f), \cdots, s_{(-x_n, -y_n)}(f)] \tag{9}$$

where $s_{(-x_n, -y_n)}(f)$ denotes the shift of the input feature maps $-x_n$ in dimension $x$ and $-y_n$ in dimension $y$ (see in

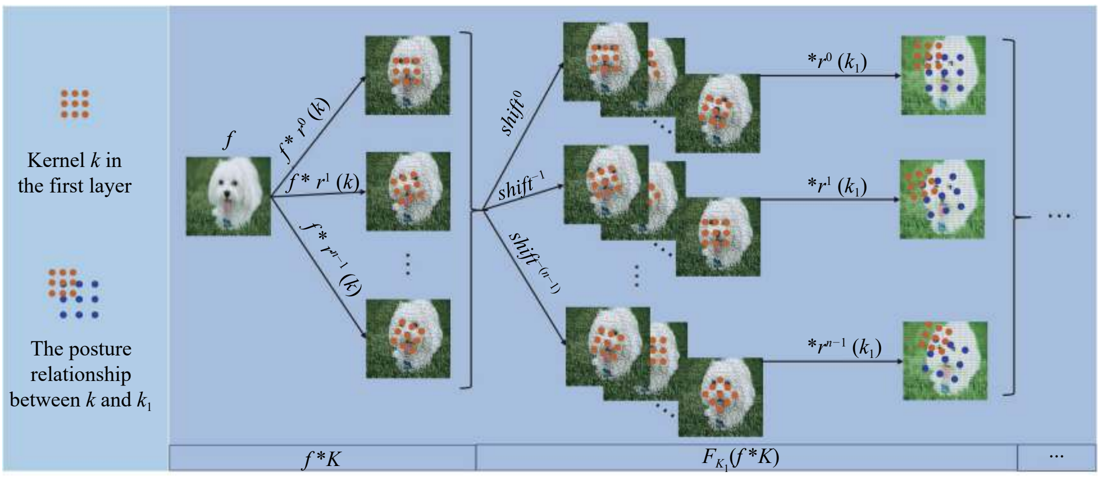

Fig. 1 Posture relationship between kernels in different layers

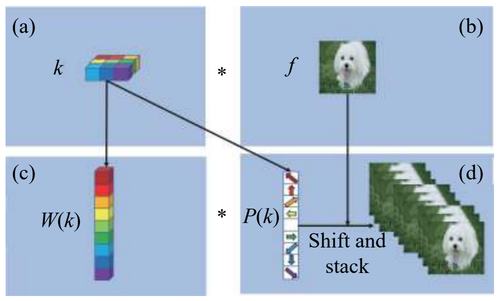

Fig. 2 Arbitrary shape convolution

Fig. 2 (d)). The element of kernel shifting $P(k)$ is equivariant to the feature maps shifting $-P(k)$. Then the convolution is (intuitive view in Fig. 2)

$$f * k = Stack_{-P(k)}(f) * W(k). \tag{10}$$

The two forms of convolution have the same parameters and computations.

When rotating a convolution kernel, its weights remain unchanged, and its locations multiply a rotation matrix:

$$\begin{aligned} W(r^i(k)) &= W(k) \\ P(r^i(k)) &= P(k)R_{g_i} \end{aligned} \tag{11}$$

where $R_{g_i}$ is a rotation matrix with degree $g_i$. Then the convolution with rotated kernel is

$$f * r^i(k) = Stack_{-P(k)R_{g_i}}(f) * W(k). \tag{12}$$

By this approach, the convolution kernel can be an arbitrary shape. At the same time, the kernel can be conveniently transformed (including rotation) by changing $P(k)$.

## 3.3 Share weights in shallow layers

In shallow layers of networks, kernels are more likely to share features in different orientations. For example, edges in different orientations can share their weights as a single kernel. Unlike previous methods which share weights in all layers, we construct networks that only share weights on shallow layers.

As shown in Fig. 3, the shallow layers are RGECs and the deeper layers are traditional convolutions. In order to induce the rotated kernel to be more sensitive to a specific direction, an orientation dimension non-maximum-suppression (NMS) loss is designed. The NMS loss will suppress the activation in all directions except the direction with the largest activation. Formally, the NMS loss is defined as the following:

$$L_{NMS} = \frac{\sum_{i=1}^{n_i}\sum_{j=1}^{n_j} \mathbf{1}(f_i^L[j] \neq \max\{f_1^L[j], \cdots, f_{n_i}^L[j]\}) f_i^L[j]}{n_i \times n_j} \tag{13}$$

where $f_i^L[j]$ is a point in the last RGEC layer (after batch normalization and ReLU operations), and $i$ indicates the rotation dimension. Notably, $n_j$ is the total number of points in the single orientation feature maps, and $n_j = h_L w_L c_L$ where $h_L$, $w_L$, and $c_L$ are the height, width, and kernel number, respectively.

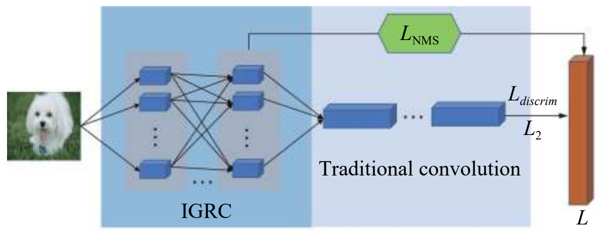

Fig. 3 Sharing weights in shallow layers

Naturally, the final loss is defined as

$$L = L_{discrim} + \lambda_1 L_{NMS} + \lambda_2 L_2 \tag{14}$$

where $L_{discrim}$ is the discriminative loss, and $\lambda_1$, $\lambda_2$ are the coefficients of the $L_{NMS}$ and $L_2$, respectively. $L_2$ is the $L_2$ regularization.

## 4 Experiments

We conducted experiments of both plain and ResNet architectures on various datasets (i.e., MNIST, CIFAR, ImageNet). Previous works with RGEC usually increase the output channels to keep the number of parameters alike. However, it will bring more computational burden. In our experiments, we keep the output channels the same as the baseline, as shown in Table 1. Therefore, no extra computations are required and the parameters and kernels are much fewer. Furthermore, to better compare the effects of RGEC, we utilized a drop-in replacement for conventional convolution.

Table 1 keeps the number of output channels in conventional convolution and RGEC the same. In Table 1, Conv is the conventional convolution, and RGEC is the rotation group equivariant convolution. $C_n$ is the number of output channels in the $n$-th layer, and we keep $C_n$ of Conv and RGEC the same. Orientation is the number of orientations, and $r$ is the orientations of RGEC that we set. $K_n$, parameters, computations are the number of kernels, parameters, and computations of the $n$-th layer, respectively. When we keep $C_n$ of Conv and RGEC the same, parameters and $K_n$ of RGEC are $\frac{1}{r}$ of Conv, and computations are the same.

## 4.1 MNIST

The MNIST database[1] of handwritten digits has a training set of 60 000 examples and a test set of 10 000

[1] http://yann.lecun.com/exdb/mnist/

Table 1 Detailed setting and resources of RGEC

| | $C_n$ | Orientation | $K_n$ | Parameters | Computations |
|---|---|---|---|---|---|
| Conv | $k_n$ | 1 | $k_n$ | $\propto k_{n-1} \times k_n$ | $\propto k_{n-1} \times k_n$ |
| RGEC | $k_n$ | $r$ | $\frac{k_n}{r}$ | $\propto \frac{k_{n-1} \times k_n}{r}$ | $\propto k_{n-1} \times k_n$ |

examples, where the digits have been size-normalized and centered in a fixed-size image. Training is performed on $28 \times 28$ images that have been shifted by up to 2 pixels in each direction with zero padding. No other augmentation/deformation is used. The models are trained using an stochastic gradient descent (SGD) optimizer with a nesterov momentum of 0.9. We set both $\lambda_1$ and $\lambda_2$ (weight decay) as 0.0001. The models have trained 200 epochs with an initial learning rate of 0.1, divided by 10 at 130, 160, and 180 epochs, respectively.

**Ordinary networks**

Firstly, we evaluate our approach SWSL (share weights on shallow layers) on MNIST via a CNN architecture. It contains 6 layers of $3 \times 3$ convolution for feature mapping, with 16, 16, 32, 32, 64, and 64 channels in each layer, respectively. Each convolution layer is followed by a batch normalization[38] and ReLU functions[13]. The max-pooling layer is followed after every 2 layers. Following the 6 feature mapping layers, a classifying layer, and a global max-pooling layer map features to 10 classes. We replace 1 to 6 layers by rotation group convolution with a basic degree of $2\pi/8$.

As shown in Table 2, SWSL($i$L) denotes that the first $i$ layers are replaced by RGECs with 8 orientations. The kernel numbers of replaced layers are in bold. The replaced layers have only 1/8 kernels compared with ordinary convolution. Due to parameter sharing, SWSL(3L) can achieve an error of 0.35% (with $L_{\text{NMS}}$), outperforming the baseline (0.40%). SWSL(3L) only has 2, 2, 4 kernels in the first three layers, while the baseline has 16, 16, 32 kernels. By sharing weights in shallow layers, SWSL(1L, 2L, and 3L) can all outperform the baseline with much fewer kernels.

**Tiny networks**

However, the results in Table 2 are based on a relatively large network. The large network may be over-parameterized, which may benefit from parameter reduction. To clarify this potential concern, we establish two tiny networks. The first tiny network has three feature mapping layers with kernels of 16, 16, and 16, respectively (Table 3). The second tiny network has 8, 16, and 32 kernels, respectively (Table 3). For these two tiny networks, the rotation group has 4 orientations. For baseline-1, SWSL(2L) with kernels 4, 4, and 16 achieves the same error of 0.63% as the baseline with kernels 16, 16, and 16. SWSL(1L) achieves 0.53% error (with $L_{NMS}$), which outperforms the baseline 0.1% error. For baseline-2, SWSL(2L) (0.57%) with kernels 2, 4, and 32 even performs slightly better than the baseline (0.61%) with kernels 8, 16, and 32. SWSL(1L) (0.52%) outperforms the baseline 0.09%. Although these two networks are small enough, SWSL can still perform better with fewer kernels. In each layer, the number of feature maps in SWSL is the same as the baseline, and the computation cost is also the same. By sharing the weights in different orientations, the rotation group layers have 1/8 (8 orientations) or 1/4 (4 orientations) kernels and parameters that are different from the layers in the baseline networks. Meanwhile, it can improve the performance with fewer parameters by sharing weights in shallow layers.

In Tables 1–3, the results with and without NMS loss of SWSL are given, respectively. The NMS loss induces kernels in each orientation to be more sensitive to a specific direction. Therefore, it can reduce redundancy in different directions. In general, SWSL with NMS loss can achieve better performances in most networks (−0.01 to 0.19 lower errors). To demonstrate that the performance increases are not from the regularization effect, we also added the NMS loss on the baseline networks in Table 3.

Table 2 Results of an ordinary network in the MNIST dataset. The kernel numbers of the replaced layers are in bold.

| Methods | #Kernels | Errors (with/without $L_{\text{NMS}}$) |
|---|---|---|
| CNN | 16, 16, 32, 32, 64, 64 | 0.40% |
| SWSL(1L) | **2**, 16, 32, 32, 64, 64 | 0.39%/0.40% |
| SWSL(2L) | **2**, **2**, 32, 32, 64, 64 | 0.38%/0.39% |
| SWSL(3L) | **2**, **2**, **4**, 32, 64, 64 | **0.35%**/0.42% |
| SWSL(4L) | **2**, **2**, **4**, **4**, 64, 64 | 0.47%/0.46% |
| SWSL(5L) | **2**, **2**, **4**, **4**, **8**, 64 | 0.42%/0.50% |
| SWSL(6L) | **2**, **2**, **4**, **4**, **8**, **8** | 0.44%/0.48% |

Table 3 Results of tiny networks on MNIST dataset

| Methods | #Kernels | Errors (with/without $L_{\text{NMS}}$) |
|---|---|---|
| CNN-tiny1 | 16, 16, 16 | 0.65%/0.63% |
| SWSL(1L) | **4**, 16, 16 | **0.53%**/0.61% |
| SWSL(2L) | **4**, **4**, 16 | 0.63%/0.66% |
| SWSL(3L) | **4**, **4**, **4** | 0.81%/1.00% |
| CNN-tiny2 | 8,16,32 | 0.72%/0.61% |
| SWSL(1L) | **2**, 16, 32 | **0.52%**/0.62% |
| SWSL(2L) | **2**, **4**, 32 | 0.57%/0.71% |
| SWSL(3L) | **2**, **4**, **8** | 0.77%/0.89% |

The NMS loss on the baseline is added on the first layer, and every 4 feature maps are regarded as a group like that in SWSL with 4 orientations. The results in Table 3 show that it cannot improve the performance of baseline networks. In conclusion, the NMS loss can improve the performance of SWSL, and the benefits do not come from regularization effects.

## 4.2 Natural image classification

We also test our method on natural image datasets such as CIFAR10, CIFAR100, and ImageNet. We replace the shallow layers with RGEC convolutions for various architectures such as VGG[14], ResNet[16], SENet[39].

CIFAR[2] dataset consists of CIFAR10 composed of 10 coarse classes and CIFAR100 composed of 20 coarse classes and 100 fine classes. Both CIFAR10 and CIFAR100 have 60 000 $32 \times 32$ color images, 50 000 and 10 000 of which are for training and testing, respectively. For both training and testing images, channel-wise normalization is applied. Each image is shifted by up to 4 pixels for each direction with zero padding and random cropped to $32 \times 32$ from the padded image or its horizontal flip.

ImageNet[3] large scale visual recognition challenge (ILSVRC) contains over 1.2 million various-size images of 1 000 classes for training and 50 000 images for validation. For both training and testing images, a channel-wise normalization is applied. For each training image, it is randomly cropped to 224×224 after randomly horizontally flipped and resized with random scales and aspect ratios. For each test image, the shorter size is resized to 256 and a $224 \times 224$ center image is cropped for evaluation. As time and computing resources are limited, we utilize 100 instead of 1 000 classes to test our approach.

**Plain networks**

The VGG16 has 16 convolution layers with kernels of 64, 64, 128, 128, 256, 256, 256, 256, 512, 512, 512, 512, 512, 512, 512, and 512, and max pooling after 2, 4, 8, 12, and 16 layers. After the 16 feature mapping layers, a final layer maps the features to 10 or 100 classes. In Table 4, SWSL(1L) to SWSL(5L) replace the first 1 to 5 layers with the rotation group convolutions with 4 orientations. We still keep the output feature maps of each layer the same between the baseline and SWSL so that the replaced layers reduce the parameters and do not increase the amount of calculation. As shown in Table 4, SWSL(3L)(6.61%) on CIFAR10 outperforms the baseline (7.35%) with a 0.74% error lower. And in CIFAR100, SWSL(5L)(26.43%) outperforms the baseline (27.51%) with a 1.08% error lower. The SWSLs of 1L to 4L on CIFAR10 have lower errors (0.33% to 0.74% lower) than the baseline, and the SWSL(5L) has slightly higher errors of 0.04%. Except that SWSL(1L) has slightly higher errors (0.05% higher), all SWSLs on CIFAR100 achieve better performances (0.20% to 1.08% lower errors).

[2] http://www.cs.toronto.edu/kriz/cifar.html

[3] http://www.image-net.org/challenges/LSVRC/

**Networks with shortcuts**

Networks on both CIFAR and ImageNet datasets are trained using SGD optimizer with nesterov momentum[40] of 0.9. We set both $\lambda_1$ (NMS loss) and $\lambda_2$ (weight decay) as 0.0001. For CIFAR, the models have trained 200 epochs with an initial learning rate of 0.1, divided by 10 at 120, 160, 180 epochs, respectively. For ImageNet, the models have trained 100 epochs with an initial learning rate of 0.1, divided by 10 at 60 and 80 epochs, respectively.

As shown in Table 5, we perform experiments on more architectures and datasets. For ResNet[16], SWSLs(1S) replace all layers (7, 11 and 37 layers for ResNet-20/32/110 on CIFAR, respectively; 5 and 7 layers for ResNet-18/34 on ImageNet, respectively) in the first stage with RGEC. The replaced layers have only 4 kernels (16 in the baseline) on CIFAR and 16 kernels (64 in the baseline) on ImageNet with 4 orientations. Between the first and second stages, a layer of $1 \times 1$ is inserted to mix the feature maps with different orientations. For fair comparisons, ResNet′ is ResNet with an extra $1 \times 1$ layer between the first and second stages, too. With the same amount of computations and 25% parameters in the first stage, SWSLs(1S) can still outperform both ResNets′ (0.10% to 0.86% lower errors) and ResNets (0.17 to 0.46% lower errors) on CIFAR. On ImageNet, SWSLs(1S) can outperform both ResNets′ (0.26% and 0.28% lower errors for ResNet-18/34′, respectively) and ResNets (0.52% and 0.36% lower errors for ResNet-18/34, respectively). For more architectures, VGG and SENet can also achieve better performances by replacing shallow layers. All the results demonstrate that shallow layers can benefit from rotational symmetry.

## 4.3 Rotated MNIST

MNIST-rot-12k[41] contains 12 000 training images and 50 000 test images, which are rotated at a random angle from the classical MNIST. For data preprocessing, we apply the channel-wise normalization and cutout operations. We use a 6-feature-mapping-layer architecture, 10 kernels (40 channels) each layer with 4 orientations. We set the dropout rate to 0.2 at the last feature-mapping layer. The model is trained by the Adam algorithm[42] with a weight decay of 0.01. The initial learning rate is 0.001 and is divided by 10 at 160 and 180 epochs (a total of 200 epochs). In Table 6, we use a similar architecture with G-CNN, and the version without NMS loss achieves a similar error rate (2.24%) with G-CNN(2.28%). Adding the proposed NMS loss ($\lambda_1 = 0.000\,1$) can achieve a lower error rate of 2.15%.

Table 4 Results of VGG16 on CIFAR datasets

| Methods | #Kernels | Errors (C10) | Errors (C100) |
|---|---|---|---|
| VGG | 64, 64, 128, 128, 256, 256, 256, 256, 512, ⋯, 512 | 7.35% | 27.51% |
| SWSL(1L) | **16**, 64, 128, 128, 256, 256, 256, 256, 512, ⋯, 512 | 6.82% | 27.56% |
| SWSL(2L) | **16, 16**, 128, 128, 256, 256, 256, 256, 512, ..., 512 | 6.79% | 27.31% |
| SWSL(3L) | **16, 16, 32**, 128, 256, 256, 256, 256, 512, ⋯, 512 | **6.61%** | 26.82% |
| SWSL(4L) | **16, 16, 32, 32**, 256, 256, 256, 256, 512, ⋯, 512 | 7.02% | 26.83% |
| SWSL(5L) | **16, 16, 32, 32, 64**, 256, 256, 256, 512, ⋯, 512 | 7.39% | **26.43%** |

Table 5 Results on ImageNet(100 classes) and CIFAR datasets

ImageNet

| Methods | #Kernels (shallow layers) | Errors |
|---|---|---|
| ResNet-18 | 64, {64, 64}×2 | 21.36% |
| ResNet-18′ | 64, {64, 64}×2 | 21.10% |
| SWSL(1S) | **16, {16, 16}**×2 | **20.84%** |
| ResNet-34 | 64, {64, 64}×3 | 20.12% |
| ResNet-34′ | 64, {64, 64}×3 | 20.04% |
| SWSL(1S) | **16, {16, 16}**×3 | **19.76%** |

CIFAR

| Methods | #Kernels (shallow layers) | Errors (C10) | Errors (C100) |
|---|---|---|---|
| ResNet-20 | 16, {16, 16}×3 | 8.03% | 32.85% |
| ResNet-20′ | 16, {16, 16}×3 | 8.43% | 32.49% |
| SWSL(1S) | **4, {4, 4}**×3 | **7.65%** | **32.39%** |
| ResNet-32 | 16, {16, 16}×5 | 7.14% | 30.85% |
| ResNet-32′ | 16, {16, 16}×5 | 7.27% | 30.75% |
| SWSL(1S) | **4, {4, 4}**×5 | **6.93%** | **30.49%** |
| ResNet-110 | 16, {16, 16}×18 | 6.02% | 27.31% |
| ResNet-110′ | 16, {16, 16}×18 | 6.52% | 27.55% |
| SWSL(1S) | **4, {4, 4}**×18 | **5.66%** | **27.14%** |
| SENet-20 | 16, {16, 16}×3 | 7.78% | 31.68% |
| SWSL(1S) | **4, {4, 4}**×3 | **7.36%** | **31.26%** |
| SENet-32 | 16, {16, 16}×5 | 7.00% | 30.04% |
| SWSL(1S) | **4, {4, 4}**×5 | **6.86%** | **29.99%** |
| SENet-110 | 16, {16, 16}×18 | 6.05% | 26.36% |
| SWSL(1S) | **4, {4, 4}**×18 | **5.59%** | **26.34%** |

Table 6 Results on MNIST-rot-12k

| Networks | Errors | Paras |
|---|---|---|
| ScatNet-2 [43] | 7.48% | – |
| PCANet-2 [44] | 7.37% | – |
| TIRBM [45] | 4.2% | – |
| ORN-8(ORNAlign) [46] | 2.25% | 0.53M |
| TI-Pooling [47] | 2.2% | 13.3M |
| CNN [9] | 5.03% | 22K |
| G-CNN [9] | 2.28% | 25K |
| Ours (without NMS) | 2.24% | 25K |
| Ours (with NMS) | **2.15%** | 25K |

## 5 Analysis

### 5.1 Visualization

To see what the rotation group convolution learns, we visualize the feature maps in baseline (in blue) and SWSL(3L) (in green) of Table 2 in Fig. 4. The first column is the input images. On the right of the input images, there are 16 feature maps of the first layer. In SWSL, each line is the feature map of a single kernel in different orientations. On the right of the feature maps, we add the feature maps with the same kernel together. As we can see, both kernels tend to find edges. This is because different orientations of a kernel are sensitive to edges of different orientations. Although both kernels tend to find edges, their emphases are still different. From the sums of feature maps of each kernel, the first one tends to find pure edges, while the second one tends to retain more information about the original image. Compared with the baseline, SWSL utilizes a single kernel to detect edges in all orientations, while the baseline utilizes many different kernels.

### 5.2 Parameter analysis

In our networks, few extra hyperparameters are needed compared with ordinary networks. For $\lambda_1$ (the weight of the NMS loss), an additional set of experiments are conducted in Table 7 (a) to show the influence of $\lambda_1$. From the results, it works well with $\lambda_1$ ranging from $10^{-5}$ to $10^{-3}$. In our experiments, though adjusting $\lambda_1$ for each model can further improve the performances, we set $\lambda_1 = 10^{-4}$ for all models. In Table 7 (b), different orientations (1 orientation is the baseline, which equals

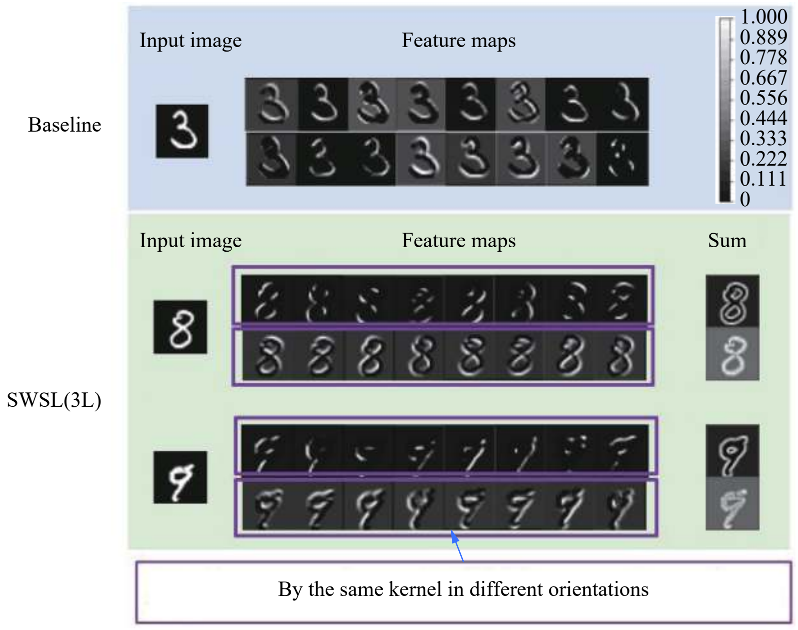

Fig. 4 Visualization: The feature maps in the first layer of the baseline and SWSL(3L) in Table 2 are visualized.

Table 7 Parameter analysis: (a) The influences of $\lambda_1$; (b) The influence of the numbers of different orientations. We utilize networks of SWSL(3L) in Table 2.

(a)

| $\lambda_1$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ | 0 |
|---|---|---|---|---|---|---|
| SWSL(3L) | 0.44% | 0.37% | 0.35% | 0.38% | 0.44% | 0.40% |

(b)

| #Orientations | 1(Baseline) | 2 | 4 | 8 |
|---|---|---|---|---|
| SWSL(3L) | 0.40% | 0.35% | 0.37% | 0.35% |

no rotation) are adopted with the same output channels in each layer. Benefiting from the weights sharing in the shallow layers, SWSLs(3L) with 2, 4 and 8 orientations all outperform the baseline.

To further study the effect of the number of output channels, we also reduce the number of kernels and keep the number of output channels the same by repeating them in Table 8. For SWSL(1L), we only replace the initial layer before the residual blocks with rotation group convolution. The first layer of SWSL(1L) only has 2 kernels with 8 orientations, and it has 16 feature maps which are the same as the baseline. To study the effect on the number of feature maps, we also test on ResNet that the first layer only has 2 kernels and repeats 8 times to 16 feature maps. The results in Table 8 show that the SWSLs(1L) of ResNet-20/32/110 achieve similar errors compared with their baselines (range from −0.18% to 0.17%). And ResNet-20/32/110 with repeat feature maps, which have the same kernels and feature maps with SWSL(1L), show obvious error increases than both SWSL(1L) (range from 0.37% to 1.22%) and the baseline (range from 0.41% to 1.39%). It demonstrates that the benefits are from weight sharing instead of the number of feature maps.

Table 8 Comparisons of ResNet with different types of the first layer

| Methods | First layer type | #Orientations | Errors (C10) | Errors (C100) |
|---|---|---|---|---|
| ResNet-20 | Normal | – | 8.03% | 32.85% |
| ResNet-20 | Repeat | – | 8.96% | 33.66% |
| SWSL(1L) | Rotate | 8 | 8.08% | 32.67% |
| ResNet-32 | Normal | – | 7.14% | 30.85% |
| ResNet-32 | Repeat | – | 7.63% | 32.03% |
| SWSL(1L) | Rotate | 8 | 7.26% | 30.81% |
| ResNet-110 | Normal | – | 6.02% | 27.31% |
| ResNet-110 | Repeat | – | 6.43% | 28.70% |
| SWSL(1L) | Rotate | 8 | 5.97% | 27.48% |

## 6 Conclusions

Based on RGEC, we have designed novel networks that share the kernel weights of different orientations in the shallow layers. Experimental results show that this approach requires much fewer kernels and parameters in the shallow layers when maintaining superior performance. It keeps the same output channels and brings no additional burden in computation. It shows that the convolution kernels in the shallow layers have rotation symmetries, and RGEC can benefit from the symmetries. Fewer kernels are more intuitive and interpretable, leading to fewer potential generalization risks.

## Appendix

### Proof of (A1)

$$r^i(f) * K = r^i(shift^i(f * K)). \tag{A1}$$

**Proof.** First, we can obtain (A2) with single kernel $k$:

$$r^i(f) * k = r^i(f) * r^i(r^{-i}(k)) = r^i(f * r^{-i}(k)). \tag{A2}$$

A kernel $k$ can be expanded to group kernels $K = [r^0(k), r^1(k), \cdots, r^{n-1}(k)]$. Define $f * K = [f * r^0(k), f * r^1(k), \cdots, f * r^{n-1}(k)]$. Then,

$$\begin{aligned}&r^i(f) * K = r^i(f * r^{-i}(K)) = \\ &r^i(f * [r^{0-i}(k), r^{1-i}(k), \cdots, r^{n-1-i}(k)]) = \\ &r^i([f * r^{0-i}(k), f * r^{1-i}(k), \cdots, f * r^{n-1-i}(k)]) = \\ &r^i(shift^i([f * r^0(k), f * r^1(k), \cdots, f * r^{n-1}(k)])) = \\ &r^i(shift^i(f * K))\end{aligned} \tag{A3}$$

where $shift^1([x_1, x_2, \cdots, x_n]) = [x_n, x_1, \cdots, x_{n-1}]$, $shift^i([x_1, x_2, \cdots, x_n]) = [x_{n-i+1}, x_{n-i+2}, \cdots, x_n, x_1, x_2, \cdots, x_{n-i}]$, and $n$ is the total number of orientations.

Notably, $r^i(x \times y) = r^i(x) \times r^i(y)$, $r^{n+i}(x) = r^i(x)$, $shift^{n+i}(x) = shift^i(x)$. Because $r^i(\cdot)$ works on the spacial dimensions and $shift^i(\cdot)$ works on the rotation dimension, they do not influence each other and $r^i(shift^j(x)) = shift^j(r^i(x))$. □

### Proof of (A4)

$$F_{K_1}(r^i(f) * K) = r^i(shift^i(F_{K_1}(f * K))). \tag{A4}$$

**Proof.**

Define $F_{K_i}(x) = [shift^0(x) * r^0(k_i), shift^{-1}(x) * r^1(k_i), \cdots, shift^{-(n-1)}(x) * r^{n-1}(k_i)]$, where $k_i$ is the weight parameter of $K_i$.

$$\begin{aligned}&F_{K_1}(r^i(f) * K) = \\ &F_{K_1}(r^i(shift^i(f * K))) = \\ &[\cdots, shift^{-(n-1)}(r^i(shift^i(f * K))) * r^{n-1}(k_1)] = \\ &[\cdots, r^i(shift^{-(n-1)+i}(f * K)) * r^{n-1}(k_1)] = \\ &r^i([\cdots, shift^{-(n-1)+i}(f * K) * r^{n-1-i}(k_1)]) = \\ &r^i([\cdots, shift^{-(n-1)}(f * K) * r^{n-1}(k_1)]) = \\ &r^i(shift^i(F_{K_1}(f * K))).\end{aligned} \tag{A5}$$

□

### Proof of (A6)

$$F_{K_m}(F_{K_{m-1}}(\cdots F_{K_1}(r^i(f) * K))) = r^i(shift^i(F_{K_m}(F_{K_{m-1}}(\cdots F_{K_1}(f * K))))). \tag{A6}$$

**Proof.**

If $F_{K_{m-1}}(\cdots F_{K_1}(r^i(f) * K)) = r^i(shift^i(F_{K_{m-1}}(\cdots F_{K_1}(f * K))))$, it can be proved easily that $F_{K_m}(\cdots F_{K_1}(r^i(f) * K)) = r^i(shift^i(F_{K_m}(\cdots F_{K_1}(f * K))))$. Denote $F_{K_{m-1}}(\cdots F_{K_1}(f * K)) = M$.

$$\begin{aligned}&F_{K_m}(F_{K_{m-1}}(\cdots F_{K_1}(r^i(f) * K))) = \\ &F_{K_m}(r^i(shift^i(M))) = \\ &[\cdots, shift^{-(n-1)}(r^i(shift^i(M))) * r^{n-1}(k_m)] = \\ &[\cdots, r^i(shift^{-(n-1)+i}(M)) * r^{n-1}(k_m)] = \\ &r^i([\cdots, shift^{-(n-1)+i}(M) * r^{n-1-i}(k_m)]) = \\ &r^i(shift^i([\cdots, shift^{-(n-1)}(M) * r^{n-1}(k_m)])) = \\ &r^i(shift^i(F_{K_m}(M))) = \\ &r^i(shift^i(F_{K_m}(F_{K_{m-1}}(\cdots F_{K_1}(f * K))))).\end{aligned} \tag{A7}$$

According to the above proof and $F_{K_1}(r^i(f) * K) = r^i(shift^i(F_{K_1}(f * K)))$ in proof of (A6), it can be deduced that $F_{K_m}(F_{K_{m-1}}(\cdots F_{K_1}(r^i(f) * K))) = r^i(shift^i(F_{K_m} \times (F_{K_{m-1}}(\cdots F_{K_1}(f * K)))))$. □

## Acknowledgements

This work was supported by National Natural Science Foundation of China (Nos. 61976209 and 62020106015), CAS International Collaboration Key Project (No. 173211KYSB20190024), and Strategic Priority Research Program of CAS (No. XDB32040000).

## References

[1] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[3] R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Columbus, USA, pp. 580–587, 2014. DOI: 10.1109/CVPR.2014.81.

[4] J. Long, E. Shelhamer, T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Boston, USA, pp. 3431–3440, 2015. DOI: 10.1109/CVPR.2015.7298965.

[5] I. D. Longstaff, J. F. Cross. A pattern recognition approach to understanding the multi-layer perception. *Pattern Recognition Letters*, vol. 5, no. 5, pp. 315–319, 1987. DOI: 10.1016/0167-8655(87)90072-9.

[6] A. G. Howard, M. L. Zhu, B. Chen, D. Kalenichenko, W.

J. Wang, T. Weyand, M. Andreetto, H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. [Online], Available: https://arxiv.org/abs/1704.04861, 2017.

[7] T. Zhang, G. J. Qi, B. Xiao, J. D. Wang. Interleaved group convolutions. In *Proceedings of IEEE International Conference on Computer Vision*, IEEE, Venice, Italy, pp. 4383–4392, 2017. DOI: 10.1109/ICCV.2017.469.

[8] X. Y. Zhang, X. Y. Zhou, M. X. Lin, J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 6848–6856, 2018. DOI: 10.1109/CVPR.2018.00716.

[9] T. Cohen, M. Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning*, New York, USA, pp. 2990–2999, 2016.

[10] M. Weiler, F. A. Hamprecht, M. Storath. Learning steerable filters for rotation equivariant CNNs. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 849–858, 2018. DOI: 10.1109/CVPR.2018.00095.

[11] M. Weiler, G. Cesa. General E(2)-equivariant steerable CNNs. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 14334–14345, 2019.

[12] Z. Y. Shen, L. S. He, Z. C. Lin, J. W. Ma. PDO-eConvs: Partial differential operator based equivariant convolutions. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8697–9706, 2020.

[13] A. Krizhevsky, I. Sutskever, G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, USA, pp. 1097–1105, 2012.

[14] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition. [Online], Available: https://arxiv.org/abs/1409.1556, 2014.

[15] C. Szegedy, W. Liu, Y. Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Boston, USA, 2015. DOI: 10.1109/CVPR.2015.7298594.

[16] K. M. He, X. Y. Zhang, S. Q. Ren, J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, USA, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[17] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Honolulu, USA, pp. 2261–2269, 2017. DOI: 10.1109/CVPR.2017.243.

[18] M. Lin, Q. Chen, S. C. Yan. Network in network. [Online], Available: https://arxiv.org/abs/1312.4400, 2014.

[19] K. M. He, X. Y. Zhang, S. Q. Ren, J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2014. DOI: 10.1109/TPAMI.2015.2389824.

[20] R. Girshick. Fast R-CNN. In *Proceedings of IEEE International Conference on Computer Vision*, IEEE, Santiago, Chile, pp. 1440–1448, 2015. DOI: 10.1109/ICCV.2015.169.

[21] S. Q. Ren, K. M. He, R. Girshick, J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Montreal, Canada, pp. 91–99, 2015.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg. SSD: Single shot MultiBox detector. In *Proceedings of the 14th European Conference on Computer Vision*, Springer, Amsterdam, The Netherlands, pp. 21–37, 2016. DOI: 10.1007/978-3-319-46448-0_2.

[23] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, USA, pp. 779–788, 2016. DOI: 10.1109/CVPR.2016.91.

[24] K. M. He, G. Gkioxari, P. Dollár, R. Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020. DOI: 10.1109/TPAMI.2018.2844175.

[25] K. Kamnitsas, C. Ledig, V. F. J. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, B. Glocker. Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, vol. 36, pp. 61–78, 2017. DOI: 10.1016/j.media.2016.10.004.

[26] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. [Online], Available: https://arxiv.org/abs/1412.7062, 2014.

[27] O. Ronneberger, P. Fischer, T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, Munich, Germany, pp. 234–241, 2015. DOI: 10.1007/978-3-319-24574-4_28.

[28] M. Ferrer, M. Gary, S. Hernández. Representation of group isomorphisms: The compact case. *Journal of Function Spaces*, vol. 2015, Article number 879414, 2015. DOI:

10.1155/2015/879414.

[29] Y. C. Xu, T. J. Xiao, J. X. Zhang, K. Y. Yang, Z. Zhang. Scale-invariant convolutional neural networks. [Online], Available: https://arxiv.org/abs/1411.6369, 2014.

[30] S. Dieleman, J. De Fauw, K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, New York, USA, pp. 1889–1898, 2016.

[31] X. Y. Cheng, Q. Qiu, A. R. Calderbank, G. Sapiro. RotDCF: Decomposition of convolutional filters for rotation-equivariant deep networks. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[32] Y. Xi, J. B. Zheng, X. X. Li, X. Y. Xu, J. C. Ren, G. Xie. SR-POD: Sample rotation based on principal-axis orientation distribution for data augmentation in deep object detection. *Cognitive Systems Research*, vol. 52, pp. 144–154, 2018. DOI: 10.1016/j.cogsys.2018.06.014.

[33] C. J. Luo, Y. Z. Zhu, L. W. Jin, Y. P. Wang. Learn to augment: Joint data augmentation and network optimization for text recognition. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Seattle, USA, pp. 13743–13752, 2020. DOI: 10.1109/CVPR42600.2020.01376.

[34] S. Graham, D. Epstein, N. Rajpoot. Dense steerable filter CNNs for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, vol. 39, no. 12, pp. 4124–4136, 2020. DOI: 10.1109/TMI.2020.3013246.

[35] M. Jacquemont, L. Antiga, T. Vuillaume, G. Silvestri, A. Benoit, P. Lambert, G. Maurin. Indexed operations for non-rectangular lattices applied to convolutional neural networks. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Prague, Czech Republic, pp. 362–371, 2019.

[36] E. Hoogeboom, J. W. T. Peters, T. S. Cohen, M. Welling. HexaConv. [Online], Available: https://arxiv.org/abs/1803.02108, 2018.

[37] C. E. Rasmussen, Z. Ghahramani. Occam's razor. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, Denver, USA, pp. 276–282, 2000.

[38] S. Ioffe, C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Lille, France, pp. 448–456, 2015.

[39] J. Hu, L. Shen, S. Albanie, G. Sun, E. H. Wu. Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011–2023, 2020. DOI: 10.1109/TPAMI.2019.2913372.

[40] I. Sutskever, J. Martens, G. Dahl, G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, USA, pp. III-1139–III-1147, 2013.

[41] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ACM, Corvalis, USA, pp. 473–480, 2007. DOI: 10.1145/1273496.1273556.

[42] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA, 2015.

[43] J. Bruna, S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013. DOI: 10.1109/TPAMI.2012.230.

[44] T. H. Chan, K. Jia, S. H. Gao, J. W. Lu, Z. N. Zeng, Y. Ma. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015. DOI: 10.1109/TIP.2015.2475625.

[45] K. Sohn, H. Lee. Learning invariant representations with local transformations. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, UK, pp.1339–1346, 2012.

[46] Y. Z. Zhou, Q. X. Ye, Q. Qiu, J. B. Jiao. Oriented response networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Honolulu, USA, pp. 4961–4970, 2017. DOI: 10.1109/CVPR.2017.527.

[47] D. Laptev, N. Savinov, J. M. Buhmann, M. Pollefeys. TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, USA, pp. 289–297, 2016. DOI: 10.1109/CVPR.2016.38.

**Zhiqiang Chen** received the B. Sc. degree in intelligence science and technology from College of Automation, University of Science and Technology Beijing, China in 2014. He received the M. Sc. and Ph. D. degrees in pattern recognition and intelligence systems with Institute of Automation, Chinese Academy of Sciences, China in 2017 and 2021, respectively. He is currently a postdoctoral fellow with Beijing Academy of Artificial Intelligence and the Institute of Automation, Chinese Academy of Sciences, China.

His research interests include machine learning, brain-inspired intelligence, medical image, and computer vision.

E-mail: chenzhiqiang2014@ia.ac.cn

ORCID iD: 0000-0003-0242-6577

**Ting-Bing Xu** received the B.Sc. degree in automation from China University of Petroleum, China in 2014, and the Ph.D. degree in computer applied technology from National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, China in 2020. He was a visiting researcher with Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Japan in 2018. He is currently a postdoctoral fellow with School of Instrumentation and Optoelectronic Engineering, Beihang University, China.

His research interests include deep learning, machine learning, pattern recognition, handwriting recognition, and computer vision.

E-mail: tingbing_xu@buaa.edu.cn

**Jinpeng Li** received the B.Eng. and M.Eng. degrees in automatic control from University of Science and Technology, China in 2012 and 2015. He received the Ph.D. degree in pattern recognition and intelligent systems from Institute of Automation, Chinese Academy of Sciences, China in 2019. He is now a researcher at Ningbo HwaMei Hospital, University of Chinese Academy of Sciences, China, and Institute of Life and Health Science, University of Chinese Academy of Sciences, China. He has authored or coauthored more than ten peer-reviewed papers in international journals and conferences.

His research interests include pattern recognition, machine learning, deep learning, transfer learning algorithms and their applications in epidemiology and medical image analysis.

E-mail: lijinpeng@ucas.ac.cn

**Huiguang He** received the B.Sc. and M.Sc. degrees from Dalian Maritime University (DMU), China in 1994 and 1997, respectively, and the Ph.D. degree (Hons.) in pattern recognition and intelligent systems from Institute of Automation, Chinese Academy of Sciences, China. From 1997 to 1999, he was an associate lecturer with DMU. From 2003 to 2004, he was a post-doctoral researcher with University of Rochester, USA. From 2014 to 2015, he was a visiting professor with University of North Carolina at Chapel Hill, USA. He is currently a full professor with Institute of Automation, Chinese Academy of Sciences. His research has been supported by several research grants from National Science Foundation of China. He has authored or co-authored more than 180 peer-reviewed papers. Dr. He is an Excellent Member of Youth Innovation Promotion Association, Chinese Academy of Sciences in 2016. He was a recipient of the Excellent Ph. D. dissertation of Chinese Academy of Sciences in 2004, the National Science and Technology Award in 2003 and 2004, the Beijing Science and Technology Award in 2002 and 2003, the K.C. Wong Education Prizes in 2007 and 2009, and the Jia-Xi Lu Young Talent Prize in 2009.

His research interests include pattern recognition, medical image processing, and brain-computer interfaces (BCI).

E-mail: huiguang.he@ia.ac.cn (Corresponding author)

ORCID iD: 0000-0002-0684-1711