Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-213051
IOS Press

OACNNs: Orientation adaptive convolutional neural networks

Xiang Ye, Zihang He, Bohan Li and Yong Li*
Beijing University of Posts and Telecommunications

Abstract. Geometric invariant feature representation plays an indispensable role in the field of image processing and computer vision. Recently, convolution neural networks (CNNs) have witnessed a great research progress, however CNNs do not excel at dealing with geometrically transformed images. Existing methods enhancing the ability of CNNs learning invariant feature representation rely partly on data augmentation or have a relatively weak generalization ability. This paper proposes orientation adaptive kernels (OA kernels) and orientation adaptive max pooling (OA max pooling) that comprise a new topological structure, orientation adaptive neural networks (OACNNs). OA kernels output the orientation feature maps which encode the orientation information of images. OA max pooling max-pools the orientation feature maps by automatically rotating the pooling windows according to their orientation. OA kernels and OA max pooling together allow for the eight orientation response of images to be computed, and then the max orientation response is obtained, which is proved to be a robust rotation invariant feature representation. OACNNs are compared with state-of-the-art methods and consistently outperform them in various experiments. OACNNs demonstrate a better generalization ability, yielding a test error rate 3.14 on the rotated images but only trained on "up-right" images, which outperforms all state-of-the-art methods by a large margin.

Keywords: Orientation adaptive kernel, rotation invariance, image transformation, feature extraction

1. Introduction

Recently, deep convolutional neural networks (CNNs) have witnessed a significant progress in various computer vision fields such as object recognition [1–5] and object detection [6–9]. CNNs are able to effectively approximate the nonlinear functions from the low-level pixel information to the high-level abstract semantic information. The pooling layers in CNNs bring the translation invariance and empower them to perform well on images that have translation/shift changes. However, CNNs lack the ability of dealing with large rotation transformations of images which limits their performance in various computer vision tasks [10–12], because in practice, images are typically taken from varying angles.

Data augmentation methods are used to improve the rotation invariance of CNNs by expanding the training dataset to include rotated images. These methods brought rotation invariance to CNNs to some extent, but the augmented training dataset is often several times larger than the original training dataset, which needs large-sized models and significantly increases the training cost.

To obtain the rotation invariance without augmenting the training dataset, Zhou et al. [13] proposed oriented response networks (ORNs) that rotated convolutional kernels by 4 or 8 angles to produce active rotating filters/kernels (ARFs). ARFs incorporated the kernels of varying orientations and hence the convolution of images with them would produce the orientation feature maps that encoded the information in images from various orientations.

However, we observed that the feature map of a particular orientation had little contribution to the representation of the semantic information of other orientations. For example, the orientation feature map at 0° contained little information useful for generating the output feature map at 45°. Thus, there may

*Corresponding author. Yong Li, Beijing University of Posts and Telecommunications. E-mail: yli@bupt.edu.cn.

be redundancy in the parameters of ARFs since they used all orientation feature maps for generating output feature maps of an individual orientation.

Also, the ORNs still employed the conventional pooling layers that partitioned the orientation feature maps into "up-right" boxes, and then aggregated the information in each "up-right" partitioned box. Experimental results showed that the "up-right" partitioning scheme dampened the orientation information extracted by the ARFs, weakening the orientation invariance of ORNs (see Section 4.1.2).

To address the parameter redundancy in ARFs, this paper proposed orientation adaptive kernels (OA kernels). A kernel of a particular orientation in the OA kernels took as the input orientation feature maps at the same orientation. The OA kernels reduced the redundant parameters, while retaining the ability of extracting orientation information. To allow for the pooling layers to extract the orientation information, we designed the orientation adaptive max pooling (OA max pooling) that partitioned the feature maps according to their orientations. We combine the OA kernels and OA max pooling to build an orientation adaptive convolutional neural network (OACNN). When trained on the "up-right" images, OACNNs achieved 96.86% accuracy on the rotated images while ORNs achieved 83.79%.

The main contributions of this paper were summarized as follows.

– The OA kernels were proposed to extract the information in images at various orientations, which contained fewer parameters than ARFs.
– We designed the OA max pooling layer that partitioned the feature maps according to their orientations. Experiment results showed that it enabled better orientation coding ability than conventional pooling layers.
– We built a network structure, OACNNs, by combining the OA kernels and OA max pooling. Various experiments demonstrated OACNNs can extract orientation invariant features from images.

2. Related works

This section reviews the existing methods that aim to learn rotation invariant feature representation, including the hand-crafted methods [14–18] and the methods of enabling CNNs to learn invariant features to rotation [19–22].

2.1. Hand-craft methods

Hand-crafted methods designed the features which were invariant to the rotation transformation of images by incorporating prior rotation knowledge. A commonly used hand-crafted feature was SIFT [14, 23] that defined a main orientation for each keypoint to extract invariant descriptors. The main orientation was computed by picking the highest local peak in gradient orientation histogram. Gradient location-orientation histogram (GLOH) [24] was an extension of SIFT descriptor designed to increase its robustness and distinctiveness. Sifting GLOH (sGLOH) [25] computed the distance of all rotation combinations by using a rotation invariant distance function, and then took the minimum as the final descriptor. The orientation filter transform (OFT) [26] was another hand-crafted feature used for the segmentation of thin structures.

The hand-crafted features were invariant to rotation transformation of images to some extent but they were mostly low-level features which did not carry sufficient high-level semantic information for object recognition, therefore they still had a limited performance in many computer vision tasks.

2.2. Convolutional neural network methods

CNNs learned task-specific feature representation through training processes that usually encoded high-level semantic information and had achieved rather intriguing performances in object recognition, image classification, etc. But currently, CNNs still suffered the problem of being unable to effectively recognize the transformed images that were unseen to them. To cope with this problem, three categories of methods had been designed for enhancing the ability of CNNs learning invariant feature representation: data augmentation methods, enhanced data augmentation methods [12, 27], and non data augmentation methods [13].

2.2.1. Data augmentation methods

Data augmentation methods firstly augmented the training dataset by including transformed images and then CNNs were trained on the original and transformed images so as to remembered both of their representation, i.e.,"learn-by-rote". [28] used orientation augmentation to improve the classification performance of CNNs. Moreover, to accurate indexing electron backscattered diffraction (EBSD) patterns, EBSD-CNN and EBSDDI-CNN integrating EBSD-CNN with dictionary indexing [29, 30] simulated EBSD patterns were generated to train CNNs, which was actually an augmentation of training dateset. Data augmentation methods brought the transformation invariance to CNNs to some extent but the original and transformed images were treated independently for training CNNs, which limited the performance.

2.2.2. *Enhanced data augmentation methods*

Unlike data augmentation methods treating the original and the transformed images independently, enhanced data augmentation methods used the augmented training dataset more effectively and reduced the redundancy in learned features by changing the network structure or adding new modules into CNNs. Dan et al. proposed multi-column networks composed of multiple branches of the same topology and trained them on the "up-right" and rotated images to obtain the rotation invariant features. Jaderberg et al. [12] proposed spatial transformer networks (STN) in which a spatial transformer module was inserted in CNNs. It learned to predict the transformation parameters of input images and then transformed them to "up-right" images which were then fed to CNNs. Laptev et al. [27] proposed a pooling operator in CNNs, transformation-invariant pooling (TI-Pooling), for learning translation invariant features. An input image was transformed by a set of pre-defined transformations and the transformed images was fed to siamese networks. The outputs of the siamese networks were then put into a TI-Pooling layer which element-wisely computed the maximum of the output features. TI-Pooling could well recognize the transformed images by any pre-defined transformation but had a limited generalization ability of recognizing the images transformed by a transformation not in the pre-defined set.

To improve the generalization ability to rotations, Fan et al. [31] proposed drop transformation network (DTN) that transform the input images by a transformation randomly picked from a set of transformations at each iteration. A dropout policy was applied to keep one output feature which was invariant to input transformations from the siamese CNNs. Cheng et al. [32] proposed rotation invariance convolutional neural network (RICNN) which inserted a rotation invariant layer in CNNs. Instead of optimizing the multinomial logistic regression objective, RICNN was trained by optimizing an objective function via imposing a regularization constraint, which explicitly enforced the feature representations of the training samples before and after rotation to be close [32].

Compared to data augmentation methods, enhanced data augmentation methods used the augmented training dataset more effectively and achieved a better performance over data augmentation methods with fewer parameters. However, the augmented training dataset was still much larger than the original training dataset and so the training cost increased. Additionally, large-sized models were also needed to well learn the feature representation on a large training dataset.

2.2.3. *Non data augmentation methods*

Both data augmentation methods and enhanced data augmentation methods brought the rotation invariance to CNNs by learning from both "up-right" and rotated images i.e., augmented datasets. As a comparison, non data augmentation methods were trained only on the "up-right" images but were able to recognize the rotated images by incorporating rotation knowledge into convolutional kernels. Zhou et al. [13] proposed oriented response networks (ORNs) which rotated conventional convolutional kernels by 4 or 8 angles to produce active rotating filters (ARFs) hard-coded the prior rotation knowledge into convolutional kernels. ARFs output orientation feature maps encoding the orientation information in images.

3. Orientation adaptive convolutional neural networks

Conventional CNNs encode the information in a scalar manner, i.e., each pixel in a conventional feature map $\mathcal{F}$ is represented by a scalar (see Table 1). The scalar manner can hardly express the information at multiple orientations in input feature maps. Different from conventional CNNs, OACNNs extract the information in input feature maps at n orientations and encode them in a n-D vector at each pixel (see $\vec{\mathcal{F}}$ in Table 1). OACNNs comprise OA kernels and OA max pooling which are designed to achieve this. The notations of formulating OACNNs are listed in Table 1.

3.1. *Orientation adaptive kernels*

Given an orientation kernel $\vec{\mathcal{K}}(\theta_0)$ at orientation θ_0, it is rotated by θ_i to generate $\vec{\mathcal{K}}(\theta_i)$, where $\theta_i = i\frac{2\pi}{n}, i = 0, 1, \ldots, n-1$. $\vec{\mathcal{K}}(\theta_i)$ encodes the information at orientation θ_i in input feature maps and n is a

Table 1
Symbol notations of formulating OACNN

Symbol	Description
C	the number of input channels
n	the number of orientations
$\theta_i = i\frac{2\pi}{n}, i = 0, 1, \ldots, n-1$	orientations
$I(\theta_0)$	"up-right" input image
$I(\theta_i)$	input image having a θ_i parallax with $I(\theta_0)$
$\mathcal{K} \in \mathbb{R}^{C\times 3\times 3}$	a conventional convolutional kernel
$\vec{\mathcal{K}}(\theta_i) \in \mathbb{R}^{C\times 3\times 3}$	an orientation kernel at θ_i
$\vec{\mathcal{K}} = [\vec{\mathcal{K}}(\theta_0), \vec{\mathcal{K}}(\theta_1), \ldots, \vec{\mathcal{K}}(\theta_{n-1})] \in \mathbb{R}^{C\times n\times 3\times 3}$	an OA kernel
$\mathcal{F} \in \mathbb{R}^{H\times W}$	a conventional feature map
$\vec{\mathcal{F}}(\theta_i) \in \mathbb{R}^{H\times W}$	an orientation feature map at θ_i
$\vec{\mathcal{F}} = [\vec{\mathcal{F}}(\theta_0), \vec{\mathcal{F}}(\theta_1), \ldots, \vec{\mathcal{F}}(\theta_{n-1})] \in \mathbb{R}^{n\times H\times W}$	an OA feature map
$\mathcal{G} = [\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{C-1}] \in \mathbb{R}^{C\times H\times W}$	conventional feature maps
$\vec{\mathcal{G}} = [\vec{\mathcal{F}}_0, \vec{\mathcal{F}}_1, \ldots, \vec{\mathcal{F}}_{C-1}] \in \mathbb{R}^{C\times n\times H\times W}$	OA feature maps

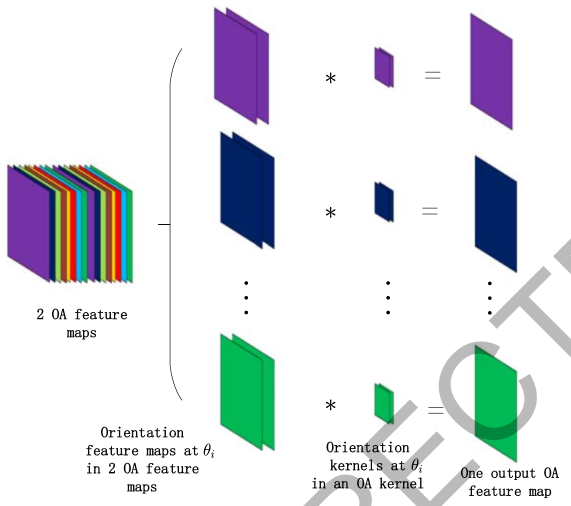
2 OA feature maps
*
=
Orientation feature maps at θ_i in 2 OA feature maps
Orientation kernels at θ_i in an OA kernel
One output OA feature map

Fig. 1. The convolution process between two OA feature maps and an OA kernel. The colors represent the orientation of feature maps or kernels.

hyper parameter representing the number of orientations that will be encoded by the OA kernels.

For each orientation convolutional layer, an OA kernel $\vec{\mathcal{K}}$ is composed of all orientation kernels, $\vec{\mathcal{K}} = [\vec{\mathcal{K}}(\theta_0), \vec{\mathcal{K}}(\theta_1), \cdots, \vec{\mathcal{K}}(\theta_{n-1})] \in \mathbb{R}^{C\times n\times 3\times 3}$, and takes the OA feature maps as an input and outputs an OA feature map encoding the information at n orientations.

3.1.1. *Obtaining $\vec{\mathcal{K}}(\theta_i)$ by rotating $\vec{\mathcal{K}}(\theta_0)$*

Given $\vec{\mathcal{K}}(\theta_0)$, let $\vec{\mathcal{K}}(\theta_i)(c, p, q)$ denote the entry of $\vec{\mathcal{K}}(\theta_i) \in \mathbb{R}^{C\times 3\times 3}$ at (c, p, q), where c is the channel index and (p, q) is the spatial coordinate. $\vec{\mathcal{K}}(\theta_i)(c, p, q)$ is computed as

$$\vec{\mathcal{K}}(\theta_i)(c, p, q) = \vec{\mathcal{K}}(\theta_0)(c, p', q'), \tag{1}$$

where

$$(p', q') = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}.$$

Note that (p', q') may not lie on integer grids and hence interpolation is needed to compute $\vec{\mathcal{K}}(\theta_0)(c, p', q')$. In this work, bilinear interpolation is employed.

3.1.2. *Forward propagation of OA kernels*

Figure 1 illustrates an example of OA feature maps of 2 channels and an OA kernel of $n = 8$ orientations and 3×3 kernel size. $\vec{\mathcal{K}}(\theta_0)$ is firstly initialized randomly with shape $2 \times 3 \times 3$ by msra [33]. Then, to construct an OA kernel $\vec{\mathcal{K}} = [\vec{\mathcal{K}}(\theta_0), \vec{\mathcal{K}}(\theta_1), \ldots, \vec{\mathcal{K}}(\theta_7)]$, $\vec{\mathcal{K}}(\theta_0)$ is rotated by Equation (1) to generate $\vec{\mathcal{K}}(\theta_i)$, $i = 1, 2 \ldots, 7$.

When the feature maps $\vec{\mathcal{G}} = [\vec{\mathcal{F}}_0, \vec{\mathcal{F}}_1] \in \mathbb{R}^{2\times 8\times H\times W}$ are convolved with $\vec{\mathcal{K}}$, the orientation kernel $\vec{\mathcal{K}}(\theta_i)$ is convolved only with the orientation feature maps at orientation θ_i. Formally,

$$\vec{\mathcal{K}}(\theta_i) * [\vec{\mathcal{F}}_0(\theta_i), \vec{\mathcal{F}}_1(\theta_i)].$$

The orientation convolution outputs an OA feature map encoding the information of input feature maps at each orientation.

3.1.3. *Back propagation of OA kernels*

Since $\vec{\mathcal{K}}(\theta_i)$, $i = 1, 2 \ldots, n-1$, are generated by rotating $\vec{\mathcal{K}}(\theta_0)$ using Equation (1), they are considered as its duplicates and hence only $\vec{\mathcal{K}}(\theta_0)$ needs to be updated. To consider the contribution of the gradient of $\vec{\mathcal{K}}(\theta_i)$, $i = 1, 2, \ldots, n-1$, to the updation of $\vec{\mathcal{K}}(\theta_0)$, we add all the gradients of $\vec{\mathcal{K}}(\theta_i)$ to update $\vec{\mathcal{K}}(\theta_0)$:

$$\delta(\theta_0) = \sum_{i=0}^{n-1} \frac{\partial L}{\partial \vec{\mathcal{K}}(\theta_i)}.$$

Then, the weights are updated by

$$\vec{\mathcal{K}}(\theta_0) = \vec{\mathcal{K}}(\theta_0) - \eta\delta(\theta_0),$$

where η is the learning rate. By updating $\vec{\mathcal{K}}(\theta_0)$, the OA kernel $\vec{\mathcal{K}}$ is updated.

3.2. *Orientation adaptive max pooling*

For $I(\theta_i)$, an orientation feature map has the strongest response at orientation θ_i, i.e., $\vec{\mathcal{F}}(\theta_i)$ typically takes the largest value. Figure 2 illustrates the orientation feature maps of $I(\theta_0)$, an "up-right" image, $I(\theta_1)$ rotating $I(\theta_0)$ by $\frac{\pi}{4}$, and $I(\theta_2)$ rotating $I(\theta_0)$ by $\frac{\pi}{2}$. Conventional max pooling partitions different feature maps with the same uniform ("up-right") rectangle grids as on $\vec{\mathcal{F}}(\theta_0)$, as shown in Figure 2(a). If it is applied to $\vec{\mathcal{F}}(\theta_1)$ in Figure 2(b), the "up-right" grid will not be aligned with $\vec{\mathcal{F}}(\theta_1)$, and hence each grid window does not extract the same feature from $\vec{\mathcal{F}}(\theta_1)$ as from $\vec{\mathcal{F}}(\theta_0)$. This dampens the orientation invariant feature extraction ability.

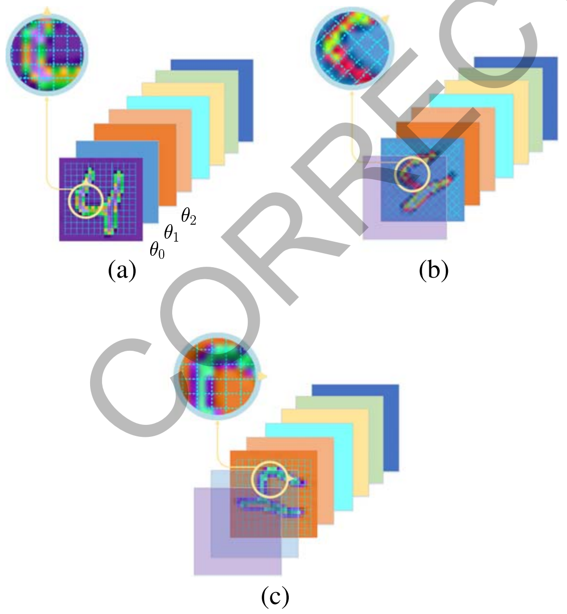

Fig. 2. A visualization of OA max pooling. (a) The input image to the CNNs is an "up-right" image $I(\theta_0)$ and its orientation feature maps at θ_0 is visualized. (b) The input image is $I(\theta_1)$, the rotated version of $I(\theta_0)$ by $\frac{\pi}{4}$ and its orientation feature maps at θ_1 is visualized. (c) The input image is $I(\theta_2)$, the rotated version of $I(\theta_0)$ by $\frac{\pi}{2}$ and its orientation feature maps at θ_2 is visualized.

To address this, we propose rotating the pooling grids according to the orientation of feature map $\vec{\mathcal{F}}(\theta_i)$. In Fig. 2(b), the grid is rotated by $\frac{\pi}{4}$ on $\vec{\mathcal{F}}(\theta_1)$. Each pooling window on $\vec{\mathcal{F}}(\theta_1)$ will enclose exactly the same content as its correspondent pooling window on $\vec{\mathcal{F}}(\theta_0)$. Likewise, Fig. 2(c) shows the rotated grid by $\frac{\pi}{2}$ on $\vec{\mathcal{F}}(\theta_2)$.

Formally, let $(h(\theta_0), w(\theta_0))^T$ denote the center of a pooling window on $\vec{\mathcal{F}}(\theta_0)$, then the center of the rotated pooling window on $\vec{\mathcal{F}}(\theta_i)$ is computed as

$$\begin{pmatrix} h(\theta_i) \\ w(\theta_i) \end{pmatrix} = \begin{pmatrix} \cos(\theta_i) & \sin(\theta_i) \\ -\sin(\theta_i) & \cos(\theta_i) \end{pmatrix} \begin{pmatrix} h(\theta_0) \\ w(\theta_0) \end{pmatrix}.$$

By this means, the pooling windows will be identical on all $\vec{\mathcal{F}}(\theta_i)$, and hence represent exactly the same position of an input image even if it is rotated.

In addition to OA max pooling, the orientation transform layer is proposed to apply an affine transformation to the learned orientation features. It increases OACNNs' representation ability on orientation information. Formally, $\vec{\mathcal{F}} = [\vec{\mathcal{F}}(\theta_0), \vec{\mathcal{F}}(\theta_1), \ldots, \vec{\mathcal{F}}(\theta_{n-1})]$ denotes an input OA feature map. Let $\mathbf{w}(\theta_0) = (a_0, a_1, \ldots, a_{n-1})$ denote the learnable transformation parameters at orientation θ_0. $\vec{\mathcal{F}}^{\text{aff}}$, the transformed OA feature map, can be written as

$$\vec{\mathcal{F}}^{\text{aff}}(\theta_j) = \mathbf{w}(\theta_j) \cdot \vec{\mathcal{F}}, \tag{2}$$

where $\mathbf{w}(\theta_j)$ is the circularly shifted $\mathbf{w}(\theta_0)$ by j, that is, $\mathbf{w}(\theta_j) = (a_{n-j}, \ldots, a_{n-1}, a_0, \ldots, a_{n-j-1})$. The orientation transform layer further improves the orientation representation ability of OACNNs (see Section 4).

3.3. *Orientation invariant feature representation*

OA kernels and OA max pooling enable the orientation information to be encoded in OA feature maps. For the purpose of obtaining rotation invariant feature representations, we can opt for three strategies: max orientation strategy, average orientation strategy, and align orientation strategy.

Given an OA feature map $\vec{\mathcal{F}} = [\vec{\mathcal{F}}(\theta_0), \vec{\mathcal{F}}(\theta_1), \ldots, \vec{\mathcal{F}}(\theta_{n-1})]$, the max orientation strategy is simply to obtain the maximum value across all orientations in it. Formally,

$$O_{\max}(p, q) = \max_{0 \le i \le n-1} \vec{\mathcal{F}}(\theta_i)(p, q).$$

Likewise, the average orientation strategy is defined to be

$$O_{\text{avg}}(p, q) = \frac{1}{n}\sum_{i=0}^{n-1} \vec{\mathcal{F}}(\theta_i)(p, q).$$

Unlike average or max orientation strategies, align orientation strategy does not reduce the number of orientation feature maps. It pushes the orientation that takes the maximum value at (p, q) to the 0^{th} dimension, and circularly pushes $\vec{\mathcal{F}}(\theta_i)$ ahead. Formally,

$$O_{\text{align}}(i, p, q) = \vec{\mathcal{F}}(\theta_{\mod ((i+m_{p,q}),n)})(p, q),$$

where $m_{p,q} = \underset{0\leq i\leq n-1}{\arg\max}\, \vec{\mathcal{F}}(\theta_i)(p, q)$. Max, average, and align orientation strategies "flatten" the orientation information output by OA kernels and OA max pooling. The rotation invariant features will be obtained when any of them is applied to OACNNs.

4. Experiment

OACNNs were evaluated on rotated image classification tasks, orientation estimation tasks, and nature image classification tasks classifying images containing various orientation information. To investigate the effectiveness of OA max pooling, the performance of OACNNs embedded with OA max pooling and ORN embedded with conventional max pooling were compared; the results showed that conventional max pooling weakened the rotation invariance ability. For each task, all the models were trained and tested for 5 times and the mean value over the 5 test results would be listed.

4.1. Rotation invariant feature representation

The rotation invariant abilities of OACNNs and the state-of-the-art methods were compared in terms of classification performance on MNIST, MNIST-rot, and MNIST-rot+. MNIST-rot was generated by rotating each image in MNIST by an angle randomly uniformly distributed in $[0, 2\pi]$. Each image sample in MNIST-rot was further rotated by eight angles, $i\frac{\pi}{4}, i = 0, \ldots, 7$, ending with MNIST-rot+ which augmented the training samples by eightfold.

A baseline CNN was employed consisting of four convolutional layers with 3×3 kernels followed by a ReLU layer and a max pooling layer, as shown in Fig. 3. To build the STN architecture, a spatial transformer layer with affine or rotation transformation was inserted as its entry layer. The structure of TI-Pooling network was generated by duplicating the baseline CNN eight times and a TI-Pooling layer was added after the duplicated CNNs but before the fully-connected layer.

For the OACNN and ORN architecture, we replaced convolutional kernels of the base CNN with OA kernels and ARFs, respectively. Three slightly different architectures were tested for OACNNs, OACNN (aff), OACNN (aff, a.s.), and OACNN (a.s.). 'aff' meant that the orientation feature maps were affine-transformed by Equation (2), which was realized by adding a orientation transform layer after the first pooling layer. 'All shared', abbreviated as a.s., meant that all the bias terms were shared in an OA kernel. To obtain the orientation invariance features, average, max, or align orientation strategy can be adopted and denoted by 'max', and 'align' in Table 2.

4.1.1. The results of recognising rotated images

As shown in the second column in Table 2, the parameter amount of TI-Pooling and STN is equal to or larger than the baseline CNN. OACNNs and ORNs both had a much smaller number of parameters than the baseline CNN, since some of their kernels were duplicated and these duplicated kernels did not need additional parameters. OACNN further reduced the parameter amount of ORNs. Among OACNN variants, the OACNN (max, a.s.) and the OACNN (avg, a.s.) have the smallest number of parameters, since their bias terms of orientation kernels were shared.

For the 'original' experiment (the column 'original' in Table 2), all the methods were trained and tested on the original MNIST dataset. OACNN (avg, aff) and OACNN (max, aff) yielded a 0.3 test error rate, and OACNN (align, aff) yielded the best result with the test error rate 0.1. They all outperformed other methods, e.g., ORNs that yielded a test error rate 0.59.

For the 'rot' experiment (the column 'rot'), all the methods were trained and tested on the MNIST-rot dataset. Training images and test images were all rotated by random degrees uniformly distributed in $[0, 2\pi]$. It was easy to see that the performance of all methods degrades on MNIST-rot in comparison with 'original'. For the baseline CNN and STNs, the degradations were more severe than ORNs and OACNNs, which indicated that OACNNs and ORNs excelled at learning rotation invariant feature representation compared with "learn by rote" methods through rotating convolutional kernels. All the OACNN variants outperformed other methods on the

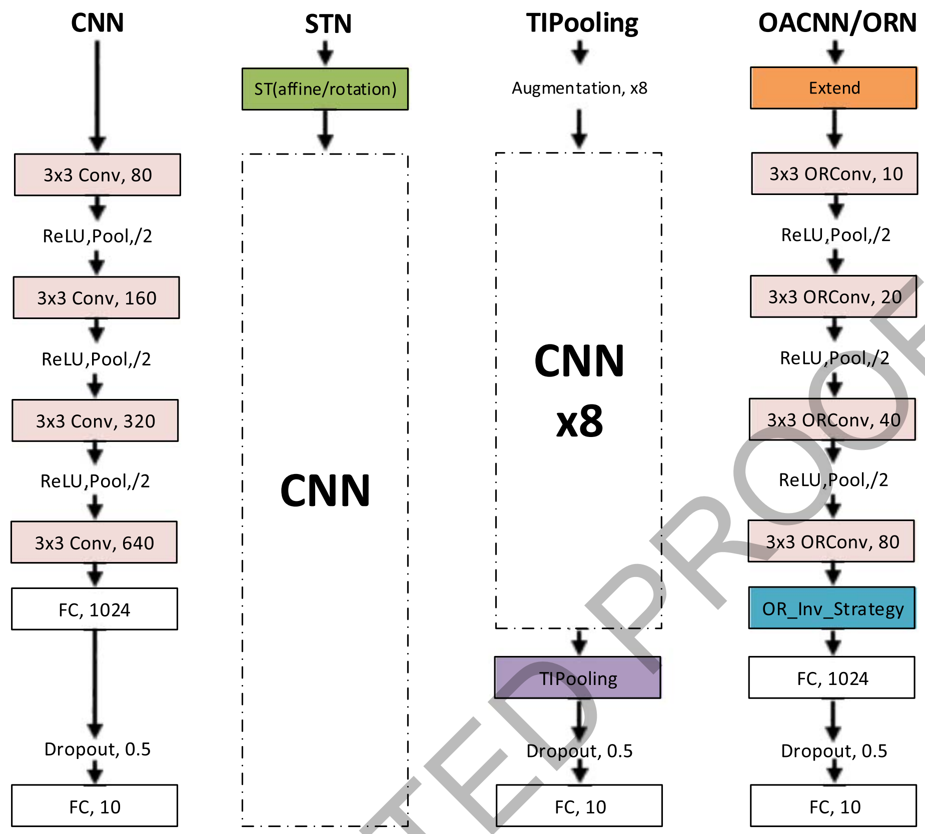

Fig. 3. Network structures. ORConv was the abbreviation of orientation convolution and OR_Inv_Strategy meant orientation invariance strategy.

Table 2

Results on the MNIST and its variants. The first column listed the evaluated models. The second column listed the percentage of the parameters of each model over that of the baseline CNN. For error rates in the third to fifth columns, each model was trained and tested on the original, the rot, and the rot+ datasets, respectively. The last column showed the error rate when the models were trained on the original training dataset but tested on the rot testing dataset

Method	params(%)	original(%)	rot(%)	rot+(%)	original → rot(%)
Baseline CNN	100.00	0.73 2.82	2.19		
STN (affine)	100.40	0.61	2.52	1.82	56.44
STN (rotation)	100.39	0.66	2.88	1.93	55.49
TI-Pooling (×8)	100.00	0.97	N/A	1.26	N/A
ORN-4 (ORPooling)	7.95	0.59	1.84	1.33	27.92
ORN-4 (ORAlign)	15.91	0.57	1.69	1.34	27.92
ORN-8 (ORPooling)	12.87	0.66	1.37	1.21	16.67
ORN-8 (ORAlign)	31.41	0.59	1.42	1.12	16.21
OACNN (max, aff)	4.28	0.30	0.90	0.99	5.40
OACNN (max, aff, a.s.)	4.27	0.65	0.88	1.04	3.40
OACNN (max, a.s.)	**4.25**	0.79	0.95	1.06	3.23
OACNN (avg, aff)	4.28	0.30	**0.80**	1.10	4.94
OACNN (avg, aff, a.s.)	4.27	0.68	1.06	1.12	3.40
OACNN (avg, a.s.)	**4.25**	0.88	1.03	1.25	3.67
OACNN (align, aff)	22.85	**0.10**	0.87	**0.91**	4.45
OACNN (align, aff, a.s.)	22.85	0.63	0.87	0.92	**3.14**
OACNN (align, a.s.)	22.82	0.79	0.92	1.08	3.34

MNIST-rot dataset, which showed the strongest orientation encoding ability, e.g., OACNN (avg, aff) yielded the best performance with the test error rate 0.80.

In the experiment of ‘rot+’, all the methods were trained and tested on the MNIST-rot+ dataset. Because of the augmentation, the performance of CNN and STNs on MNIST-rot+ was improved over MNIST-rot. However, they still had inferior performances to OACNNs and ORNs. All the OACNNs outperformed the other methods, except OACNN (avg, a.s) on MNIST-rot+ dataset. OACNN (align, aff) yielded a test error rate 0.91, which was the best performance on MNIST-rot+.

Next, we investigated the generalization of different methods in the experiment ‘original $\rightarrow$ rot’. All the methods were trained on the original MNIST training dataset, and tested on the MNIST-rot test dataset. Training samples only included “up-right” images while test samples included rotated images. From Table 2, CNN and STNs yielded 56.28, 56.44, and 55.49 test error rates respectively. This indicated CNN and STNs trained only on “up-right” images could only recognise a few rotated images correctly, i.e. had a weak rotation invariance ability.

As a comparison, OACNNs and ORNs equipped with rotated kernel OA kernels or ARFs demonstrated a much stronger ability of learning rotation invariant features. OACNNs performed better than ORNs. In Table 2, OACNN (align, aff, a.s.) yielded a test error rate 3.14 in ‘original $\rightarrow$ rot’ experiment, which was even comparable to that of STNs or TI-Poolings on the dataset MNIST-rot and MNIST-rot+ containing augmented data.

4.1.2. *The influence of image size on rotation invariance*

As mentioned in Section 3.2, conventional pooling grids on orientation feature maps were not aligned with each other. This disalignment dampened the rotation invariant performance at some input image sizes. In this section, the rotation invariant performance was compared on the ‘original $\rightarrow$ rot’ experiment, with 32×32 and 31×31 input sizes.

As shown in Table 3, while the performance of ORNs decreases when the input image size changed from 32 to 31, OACNNs yielded a comparable performance before and after the image size change. OA max pooling always is able to align the grids with the orientation feature maps, and enables the OACNNs to be robust against the input image size.

Table 3
A comparison of rotation invariance performance with different input sizes, 32×32 and 31×31. OACNNs embedded with OA max pooling and ORNs embedded with conventional max pooling were compared and the test error rates were listed.

Method	32×32	31×31
ORN-4 (ORPooling)	27.92	34.43
ORN-4 (ORAlign)	27.92	34.22
ORN-8 (ORPooling)	16.67	25.43
ORN-8 (ORAlign)	16.21	25.35
OACNN (max, aff)	5.40	6.55
OACNN (max, aff, a.s.)	3.40	3.87
OACNN (max, a.s.)	3.23	4.10
OACNN (avg, aff)	4.94	4.51
OACNN (avg, aff, a.s.)	3.40	3.09
OACNN (avg, a.s.)	3.67	4.02
OACNN (align, aff)	4.45	5.98
OACNN (align, aff, a.s.)	3.14	**3.72**
OACNN (align, a.s.)	3.34	4.24

4.2. *Orientation estimation*

To evaluate the orientation encoding ability of OACNNs, ORNs, and baseline CNNs, their performances of estimating the orientations of images were compared. Images in MNIST dataset were all regarded as “up-right” images, i.e. the orientation of images was equal to 0 and the orientation of a rotated image was defined to be its rotation angle. All methods were trained to predict the orientation of images. To this end, the orientation invariant strategies were removed from OACNNs, since the orientation was to be estimated.

Images in MNIST dataset were rotated by $i\frac{2\pi}{N}$ to generate dataset MNIST-N where $i \in [0, N)$ was a random integer and $N = 8, 16$ or 32. As shown in Table 4, on MNIST-8, OACNNs and ORN got much better performance than baseline CNN. OACNNs outperformed ORN. In particular, OACNN (a.s.) yielded the best performance with a test error rate 0.01. On MNIST-16, OACNNs and ORNs again outperformed the baseline CNN. ORN-8 yielded the best performance with a 0.01 test error rate. OACNNs got a relatively inferior performance to ORNs on MNIST-16. On MNIST-32, OACNNs and ORNs yielded better performance than CNNs. OACNN (aff) got the best performance with a test error rate 0.37. Noticed that, as was discussed in Section 3.1 and Table 2, the parameter amount of OACNNs was only $\frac{1}{3}$ that of ORNs.

4.3. *Natural image classification*

CIFAR-10 and CIFAR-100 included 32×32 nature images with 10 and 100 classes of objects,

Table 4
Predicting the orientations of images. The test error rates were listed

Method	MNIST-8	MNIST-16	MNIST-32
Baseline CNN	75.0	1.04	0.76
ORN-4	N/A	0.11	0.43
ORN-8	3.60	**0.01**	0.39
OACNN (aff)	1.60	0.68	**0.37**
OACNN (aff, a.s.)	0.28	0.79	0.44
OACNN (a.s.)	**0.01**	0.72	0.49

Table 5
Nature image classification. The test error rates were listed

Method	CIFAR-10	CIFAR-100
CNN	20.49	54.49
ORN-4(ORPooling)	22.45	58.62
ORN-4(ORAlign)	22.36	58.27
ORN-8(ORPooling)	21.46	57,86
ORN-8(ORAlign)	23.07	59.99
OACNN	**19.69**	**50.96**
VGG	6.32	28.49
VGG_ORG	5.47	27.03
VGG_OACNN	**5.38**	**26.53**

respectively. Two backbones were employed, a four-layer backbone used in Section 4.1 and VGGNet [2] which were notated by 'VGG' in Table 5, e.g., VGG_OACNN.

As shown in Table 5, with the four-layer backbone, OACNNs yielded test error rates 19.69 and 50.96, which were the best performance on CIFAR-10 and CIFAR-100, respectively. When a larger model was applied, VGG_OACNN again yielded the best performance with test error rates 5.38 and 26.53. With stronger orientation encoding ability of OA kernels and OA max pooling in OACNNs, they were effective to encode orientation informations in nature images.

5. Conclusion

This work proposed OACNNs that enabled CNNs to extract orientation invariant features even without data augmentation. The OACNNs comprised the OA kernels and OA max pooling layers. By rotating the convolutional kernels, the OA kernels could extract the information in an image at multiple orientations and thus encoded the orientation information of the image. In addition, the designed pooling scheme allowed for the OA max pooling layer to further retain the orientation information in the orientation feature maps.

Experiments showed that OACNNs outperformed the state-of-the-art methods in extracting orientation invariant features. The effectiveness of OA kernels demonstrated that the reduction of parameter amount compared to ARFs did not decrease the accuracy. Thus feature map of a particular orientation had little contribution to the representation of the semantic information of other orientations. Moreover, previous works did not consider the pooling layers in CNNs for designing orientation invariant architectures, the effectiveness of OA max pooling demonstrated that it was helpful to consider effect of pooling layers on the rotation invariance of CNNs.

The proposed OACNNs currently consider only 8 discrete predefined orientations. When the input images undergo a continuous rotation (e.g., 10°), the ability of extracting invariant features may deteriorate, e.g., the performance decreased on the "original $\rightarrow$ rot" experiment in comparison with the "original" experiment (see Table 2). One future work is to design the OA kernels that contain more orientations to address the images undergoing continuous rotation degrees.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62071060), the National Key R&D Program of China under Grant No. 2019YFF0302601, and the Beijing Key Laboratory of Work Safety and Intelligent Monitoring Foundation.

References

[1] A. Krizhevsky, I. Sutskever and G.E. Hinton, Imagenet classification with deep convolutional neural networks, In *NIPS* **1** (2012), 4.
[2] Karen Simonyan and Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, In *Computer Vision and Pattern Recognition*, 10 2014.
[3] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *IVPR*, pages 770–778, 2016.
[4] Yanteng Zhang, Qizhi Teng, Linbo Qing, Yan Liu and Xiaohai He, Lightweight deep residual network for alzheimer's disease classification using smri slices, *Journal of Intelligent and Fuzzy Systems*, 2021.
[5] Runze Song, Zhaohui Liu and Chao Wang, End-to-end dehazing of traffic sign images using reformulated atmospheric scattering model, *Journal of Intelligent and Fuzzy Systems*, 2021.
[6] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, In *Computer Vision and Pattern Recognition*, 10 2014.

[7] Shaoqing Ren, Kaiming He, Ross B. Girshick and Jian Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2015.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C. Berg, Ssd: Single shot multibox detector, In *ECCV*, 12 2016.

[9] Hei Law and Jia Deng, Cornernet: Detecting objects as paired keypoints, In *ECCV*, 8 2018.

[10] Sam Hallman and Charless C. Fowlkes, Oriented edge forests for boundary detection, In *Computer Vision and Pattern Recognition*, pages 1732–1740, 2015.

[11] Gong Cheng, Peicheng Zhou and Junwei Han, Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection, In *Computer Vision and Pattern Recognition*, pages 2884–2893, 2016.

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman and Koray Kavukcuoglu, Spatial transformer networks, In *Computer Vision and Pattern Recognition*, 4 2016.

[13] Yanzhao Zhou, Qixiang Ye, Qiang Qiu and Jianbin Jiao, Orientated response networks, In *Computer Vision and Pattern Recognition*, pages 4961–4970, Honolulu, Hawaii, USA, 2017.

[14] David G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, pages 91–110, 12 2004.

[15] Cordelia Schmid and Jean Ponce, Semi-local affine parts for object recognition, *Proc.british Machine Vision Conf* **2** (2010), 959–968.

[16] Muhammed Jamshed, Shahnaj Parvin and Subrina Akter, Significant hog-histogram of oriented gradient feature selection for human detection, *International Journal of Computer Applications* 132, 2015.

[17] Yucel Uzun, Muhammet Balcilar, Khudaydad Mahmoodi, Feruz Davletov, M. Fatih Amasyali and Sirma Yavuz, Usage of hog (histograms of oriented gradients) features for victim detection at disaster areas, In *International Conference on Electrical and Electronics Engineering*, pages 535–538, 2013.

[18] Ping Shu Ge, Guo Kai Xu, Xiu Chun Zhao, Peng Song and Lie Guo, Pedestrian detection based on histograms of oriented gradients in roi, *Advanced Materials Research* **542-543** (2012), 937–940.

[19] Xu Shen, Xinmei Tian, Anfeng He, Shaoyan Sun and Dacheng Tao, Transform-invariant convolutional neural networks for image classification and search, In *ACM on Multimedia Conference*, pages 1345–1354, 2016.

[20] Sergey Demyanov, James Bailey, Ramamohanarao Kotagiri and Christopher Leckie, Invariant backpropagation: how to train a transformation-invariant neural network, *Computer Science*, 2016.

[21] A. Ravichandran and B. Yegnanarayana, A two-stage neural network for translation, rotation and size-invariant visual pattern recognition, In *International Conference on Acoustics, Speech and Signal Processing* **4**(1991), 2393–2396.

[22] R. Southworth, K.V. Mardia and C.C. Taylor, Transformation- and label-invariant neural network for the classification of landmark data, *Journal of Applied Statistics* **27**(2) (2000), 205–215.

[23] Tak W. Yan and Hector Garcia-Molina, Sift: a tool for wide-area information dissemination, In *Usenix Technical Conference*, 1995.

[24] K. Mikolajczyk and C. Schmid, A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(10) (2005), 1615–1630.

[25] Fabio Bellavia, Domenico Tegolo and Emanuele Trucco, Improving sift-based descriptors stability to rotations, In *2010 20th International Conference on Pattern Recognition*, pages 3460–3463, 2010.

[26] K. Sandberg and M. Brega, Segmentation of thin structures in electron micrographs using orientation fields, *Journal of Structural Biology* **157**(2) (2007), 403–415.

[27] Dmitry Laptev, Nikolay Savinov, Joachim M. Buhmann and Marc Pollefeys, Ti-pooling:transformation-invariant pooling for feature learning in covolutional neural networks, In *Computer Vision and Pattern Recognition*, 12 2016.

[28] Luke Taylor and Geoff Nitschke, Improving deep learning using generic data augmentation, *CoRR*, abs/1708.06020, 2017.

[29] Z. Ding, E. Pascal and M. De Graef, Indexing of electron back-scatter diffraction patterns using a convolutional neural network, *Acta Materialia* **199** (2020), 370–382.

[30] Zihao Ding, Chaoyi Zhu and Marc De Graef, Determining crystallographic orientation via hybrid convolutional neural network, *Materials Characterization* **178** (2021), 111213.

[31] Chunxiao Fan, Yang Li, Guijin Wang and Yong Li, Learning transformation-invariant representations for image recognition with drop transformation networks, **6** (2018), 73357–73369.

[32] Gong Cheng, Peicheng Zhou and Junwei Han, Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images, *IEEE Transactions on Geoscience and Remote Sensing* **54**(12) (2016), 7405–7415.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *CoRR*, abs/1502.01852, 2015.