RRL: Regional Rotate Layer in Convolutional Neural Networks

Zongbo Hao, Tao Zhang, Mingwang Chen, Zou Kaixu
University of Electronic Science and Technology of China
No.4, Section 2, North Jianshe Road
Chengdu, China, 610054
zbhao@uestc.edu.cn, zhangtao897476472@126.com, chenmingwang815@163.com, 1241510971@qq.com

Abstract

Convolutional Neural Networks (CNNs) perform very well in image classification and object detection in recent years, but even the most advanced models have limited rotation invariance. Known solutions include the enhancement of training data and the increase of rotation invariance by globally merging the rotation equivariant features. These methods either increase the workload of training or increase the number of model parameters. To address this problem, this paper proposes a module that can be inserted into the existing networks, and directly incorporates the rotation invariance into the feature extraction layers of the CNNs. This module does not have learnable parameters and will not increase the complexity of the model. At the same time, only by training the upright data, it can perform well on the rotated testing set. These advantages will be suitable for fields such as biomedicine and astronomy where it is difficult to obtain upright samples or the target has no directionality. Evaluate our module with LeNet-5, ResNet-18 and tiny-yolov3, we get impressive results.

Introduction

Deep learning and convolutional neural networks have made great progress in many tasks such as image classification and object detection. The inherent properties of convolution and pooling layer alleviate the influence of local translation and distortion. However, due to the lack of the ability to process large rotation of image, convolution neural networks are limited in some visual tasks, including target boundary detection (Maninis et al. 2016; Dalal and Triggs 2005), multi-directional target detection (Cheng, Zhou, and Han 2016) and image classification (Jaderberg et al. 2015; Laptev et al. 2016). In recent years, CNN based image classification and object detection have been used in biomedical, industrial and astronomical research. In these fields, objects can appear in any direction, such as microscopic images, objects on conveyor belts or objects observed. So, the research on rotation invariance of neural networks has been more and more important.

At present, most deep networks use data augmentation to make the network recognize objects in different directions (Ojala, Pietikäinen, and Harwood 1996; Quiroga et al. 2018; Tamura, Horiguchi, and Murakami 2019; Mash, Borghetti, and Pecarina 2016), or merging the rotation equivariant features (Gao et al. 2019; Wiersma, Eisemann, and Hildebrandt 2020). These methods either increase the workload of training or increase the number of model parameters. In this paper, by making the feature maps before and after convolution rotation equivariant, the whole neural network is rotation invariant. With rotation angle $\theta \in \{0, 90, 180, 270\}^\circ$, the feature maps are completely the same. When the input image is rotated with arbitrary angle, there is only small difference between feature maps before and after rotation.

The main contributions of this paper are:

1) A Local Binary Pattern(*LBP*) operator based Regional Rotation Layer (*RRL*) is proposed. *RRL* can be embedded in CNNs, without the need for substantial changes to the network to achieve rotation invariant.

2) Without learning new parameters, *RRL* makes the feature maps before and after convolution satisfy the rotation equivariance, and thus makes the entire neural network rotation invariant. With rotation angles $\theta \in \{0, 90, 180, 270\}^\circ$, the feature maps are exactly the same. With arbitrary rotation angle, there is a small distinction between feature maps.

3) Evaluate *RRL* with LeNet-5, ResNet-18 and tiny-yolov3, we get impressive results.

Related Work

For deep learning based methods, the most direct way is data augmentation (Van Dyk and Meng 2001), which simply changes the size and direction of the input images to create more training data. TI-Pooling (Laptev et al. 2016) uses the rotated image as input, and applies a pooling layer before outputting features to unify the network's results for different rotation angles. Dieleman proposed a deep neural network model that uses translational and rotational symmetry to classify galaxy morphology (Dieleman, Willett, and Dambre 2015). They create multiple rotated and flipped galaxy image samples, and then concatenate the feature maps to the classifier. Polar Transformer Networks (PTN) (Esteves et al. 2018) converts the input to polar coordinates. PTN is composed of a polar coordinate prediction module, a polar coordinate conversion module, and a classifier to achieve translation invariance and equal changes in expansion/rotation groups. (Jiang and Mei 2019) also proposed a Polar Coordinate Convolutional Neural Network (PCCNN) to convert the input image to polar coordinates to achieve rotation invariance. The overall structure of the model is the same as the traditional CNN, except that the central loss function is used to learn rotation invariant features. In addition, (Cohen and Welling 2016) proposed Group Equivariant Convolutional Networks (GCNN) as a special case of controllable CNN, which proved that the spatial transformation of the image can be corresponded in the feature map and the filter. GCNN is composed of group convolution kernels. These convolutions include filter rotation and merging operations on the rotation. Group convolution is limited to integer multiples of 90°rotation and flipping. Cohen et al. also proposed steerable CNNs (Cohen and Welling 2017) and Spherical CNNs (Cohen et al. 2018) to achieve rotation equivariant. Steerable CNNs are limited to discrete groups, such as discrete rotations acting on planar images or permutations acting on point clouds. Spherical CNNs show good robustness. Because FFT and IFFT are used in spherical convolution, some information will be lost in the conversion process. Spherical convolution achieves rotation invariance for ideal 3D objects, and there is no interference of background or other noise. If there are multiple 3D objects in the natural scene, the 3D objects must be segmented first, and then the rotation invariant features are extracted. The Rotation Equivariant Vector Field Networks (RotEqNet) (Marcos et al. 2017) uses multiple rotation instances of a uniform standard filter to perform convolution, that is, the filter is rotated at different intervals. Although the RotEqNet model is small, the increasing in the number of convolution kernels brings more memory and longer computing time. (Dieleman, De Fauw, and Kavukcuoglu 2016) encodes cyclic symmetry in CNNs by parameter sharing to achieve rotation equivariant. They introduce four operations: slice, pool, stack and roll. The operations can be cast as layers in a neural network, and build networks that are equivariant to cyclic rotations and share parameters across different orientations. But the operations change the size of the minibatch (slicing, pooling), the number of feature maps (rolling), or both (stacking). To alleviate the excessive time-consuming and memory usage, (Li et al. 2018) proposed Deep Rotation Equivariant Network. They apply rotation transformation on filters rather than feature maps, achieving a speed up of more than 2 times with even less memory overhead. But the methods all need to learn new parameters to achieve rotation equivariant.

Rotation Invariance Based on LBP Operator

The standard convolutional neural networks do not have the property of rotation invariance. Trained by the upright samples, the performance drops significantly when tested by the rotated images. To solve this problem, we add a regional rotation layer (*RRL*) before the convolutional layers and the fully connected layers. The main idea is that we indirectly achieve rotation invariance by restricting the convolutional features to be rotation equivariant.

Local Binary Pattern

Local Binary Pattern (*LBP*) (Ojala, Pietikäinen, and Harwood 1996) is an operator that describes image texture features. Suppose the window size is 3 × 3, setting the central pixel as the threshold, we can get a binary encoding of the local texture, and convert it to a decimal value. As shown in Figure 1(a), the central point pixel value 6 is used as the comparison reference, then calculate the difference values of the surrounding eight pixels with the central point. If the neighbouring value is less than the central value, the corresponding location is marked as 0, otherwise marked as 1, as shown in Figure 1(b). Taking the upper left corner of the matrix as the starting point, each position is given an index power of 2 according to the flattening and stretching direction of the matrix, as shown in Figure 1(c). The dot production operation is performed between the weight matrix and the binary matrix, as shown in Figure 1(d). Only the values of 1 in the binary matrix are preserved, and the new weight is superimposed. Finally, the surrounding elements of the result matrix are added to form the decimal LBP identifier (in this example 169) of the local texture. A series of LBP feature values are obtained by rotating the surrounding points, and the minimum of these values is selected as the LBP value of the central pixel. In this paper, the points are rotated to the minimum state of LBP, so as to achieve the rotation invariance of angle. In the case, the minimum state is shown as Figure 1 (e). That is, the original feature is rotated 135°clockwise, as shown in Figure 1(f).

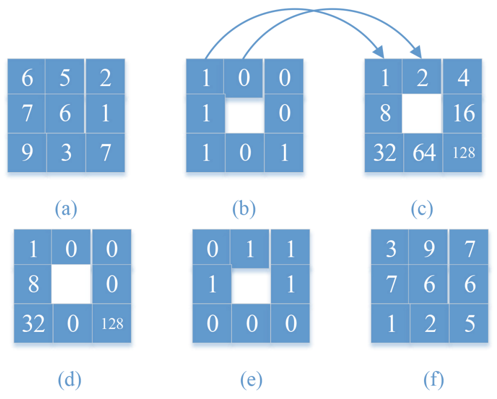

Figure 1: Local Binary Pattern. (a) image of size 33. The numbers are the gray values; (b) the binary values of the surrounding pixels; (c) the index value of the surrounding pixels; (d) the dot product result of (b) and (c); (e) rotate (b) of 135°clockwise to get the minimum LBP; (f) the image window after rotation.

.

Regional Rotation Layer (*LBP*)

LBP is operated in a window, while *RRL* is operated on the feature maps. The feature maps are sampled one by one in the form of sliding window, and LBP is implemented in each window. So we can rotate the feature maps to the same state even with different input orientations.

RRL is usually added before convolutional layer. Here we take the first convolution operation of a three-channel RGB image as an example to illustrate the workflow of *RRL*.

```
Algorithm 1: RRL in local windows
Input: RGB image sample batch $\{I_1, I_2, \cdots, I_t \cdots\}$
Output: Rotate the feature maps to the same state
1: Load image $I_t$, $I_t \in \mathbb{R}^{H \times W \times 3}$.
2: Perform LBP on each channel to get $V_t$, $V_t \in \mathbb{R}^{F \times F \times (O_H \times O_W \times 3)}$, where $F$ is the kernel size, $O_{H(W)}$ is the height/width of the feature map. The sequence of feature maps in the channel has not been changed, shown as process ① in Figure 2.
3: Reshape $V_t$, then concatenate the window features belonging to the same channel into a matrix $I'_t$, $I'_t \in \mathbb{R}^{(F \times O_H) \times (F \times O_W) \times 3}$, shown as process ② in Figure 2.
4: Perform a convolution operation with step size $F$ on $I'_t$ and get the output feature $O_t$, $O_t \in \mathbb{R}^{O_H \times O_W \times k'}$, shown as process ③ in Figure 2.
```

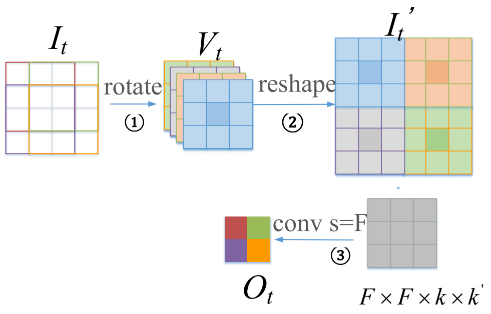

Figure 2: *RRL* works in a local window I_t. Step ①: Perform *LBP* on each channel to rotate feature maps. Step ②: Reshape V_t and concatenate the features into matrix I'_t. Step ③: Perform convolution operation on I'_t to get output feature Q_t.

Rotation Equivariance and Invariance Derivation

Equivariance refers to that, when the transformation can be measured in the output of an operator, the operator and the transformation are equivariant, as shown in eq. 1.

$$f(T(x)) = T(f(x)) \tag{1}$$

where x is the input, $T(\bullet)$ is the transformation, $f(\bullet)$ is the operator.

An operator is invariant relative to the transformation, when the influence of the transformation cannot be detected in the output of the operator, shown as eq. 2.

$$f(T(x)) = f(x) \tag{2}$$

In order to achieve rotation invariance of the entire CNN, it is expected that the input features can be rotated uniformly after the convolution layers and before the fully connected layers that perform the classification task.

Local convolution operation is equivariant. In the feature window $w(F \times F)$, when w is rotated of r (r means counterclockwise rotate 90°, and r^n means counterclockwise rotate n*90°), if the convolution kernel rotates the same angle r, the result is unchanged: $f(w) = L_r[f](rw)$, where $L_r[f]$ indicates that the convolution kernel rotates r counterclockwise. But the local convolution operation is not invariance, when the convolution kernel is unchanged, the results before and after w rotation are different: $f(w) \neq f(rw)$.

Global convolution operation has neither rotation equivariance nor invariance. When the feature map is rotated, not only the convolution kernel needs to be rotated to the same angle, but the feature output must be rotated with the same angle in the opposite direction, so that the result of the original convolution operation can be kept consistent, as shown in eq. 3.

$$f(x) = r^{-1} L_r[f](rx) \tag{3}$$

From the above analysis, we know that after rotation (only for r^n rotation), the features will still maintain equivariant after layer-by-layer convolution. Therefore, the entire CNN will be rotation invariant if we reversely rotate the feature maps before the fully connected layer.

First, to achieve equivariant of global convolution, the core function $R(x)$ of the algorithm needs to satisfy eq. 4:

$$f^F[R(x)] = r^{-n} f^F[R(r^n x)] \tag{4}$$

where $f^F(x)$ is the convolution operation with step size of F. In other words, when the filter does not change, the convolution result of the rotated input is equal to that of the non-rotated input through reverse rotating the output. To satisfy eq. 4, the local convolution needs to be invariant: $f[R(w)] = f[R(r^n w)]$. That is :

$$R(w) = R(r^n w) \tag{5}$$

Here, we use the core function $R(x)$, which is named *RRL* module, to make the window convolution invariant, and achieve rotation invariant of the CNN. *RRL*s position is before each convolutional layer and after the last convolutional layer with the step size of F. In particular, for the last *RRL*, the global feature maps x are treated as a local window w, and satisfies $R(x) = R(r^n x)$. Because the activation function, BN layer and pooling layer are rotation equivariant and they do not affect the final result, they are not discussed here.

Integrate RRL with CNN

Each *RRL* R_i is embedded before each conv layer f_i. Suppose that the original feature x and the rotated 90°feature rx are fed into R_1 respectively.

After R_1 and f_1, we have:

$$\begin{aligned} x &\to f_1^{F_1}[R_1(x)] \\ rx &\to f_1^{F_1}[R_1(rx)] \\ \therefore f_1^{F_1}[R_1(x)] &= r^{-1} f_1^{F_1}[R_1(rx)] \end{aligned}$$

After R_2 and f_2, we have:

$$\begin{aligned} &f_1^{F_1}[R_1(x)] \to f_2^{F_2}\left[R_2\left[f_1^{F_1}[R_1(x)]\right]\right] \\ &f_1^{F_1}[R_1(rx)] \to f_2^{F_2}\left[R_2\left[f_1^{F_1}[R_1(rx)]\right]\right] \\ &\therefore f_2^{F_2}\left[R_2\left[f_1^{F_1}[R_1(x)]\right]\right] = r^{-1} f_2^{F_2}\left[R_2\left[f_1^{F_1}[R_1(rx)]\right]\right] \end{aligned}$$

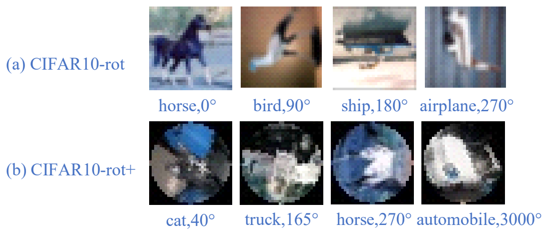

Figure 3: Examples of CIFAR10-rot and CIFAR10-rot+

.

So, we have:

$$R_{n+1}\left[f_n^{F_n}\left[\dots\left[R_2\left[f_1^{F_1}\left[R_1\left(x\right)\right]\right]\right]\right]\right] \\ = R_{n+1}\left[f_n^{F_n}\left[\dots\left[R_2\left[f_1^{F_1}\left[R_1\left(rx\right)\right]\right]\right]\right]\right] \tag{6}$$

The conclusion can be extended to other CNNs. When *RRL* is added in the right position, the rotation invariance of the model can be achieved.

Experiments

Image Classification Based on LeNet-5

Dataset and LeNet-5 CIFAR-10 is used in our experiment. The dataset was proposed by krizhevsky in 2009. It contains 60000 32 $\times$ 32 colour images, belonging to 10 categories. There are 50000 images in training set (5000 in each category) and 10000 images in test set (1000 in each category). The images rotated in the first ways are call CIFAR10-rot (namely $\theta \in \{0, 90, 180, 270\}°$and in the second way are called CIFAR10-rot+ ($\theta \in [0, 360)°$), as shown in Figure 3. In order to ensure that the effective content area of the image is fixed, the largest inscribed circle of the square image is selected. Only the inner area of the circle has original image pixels, and the outer area of the circle is filled with black, shown as Figure 3 (b).

LeNet-5 (LeCun et al. 1998) is one of the earliest CNNs. It has two convolutional layers and three fully connected layers, so three RRLs are plugged in. Keeping the convolutional layers and fully connected layers unchanged as (LeCun et al. 1998), the three RRLs are inserted in front of conv1, conv2 and after conv2, respectively.

Experimental Result and Analysis Table 1 shows the test accuracy of LeNet-5 on CIFAR-10 with and without *RRL*. The second column is trained by the original training set (without rotation images) and tested by CIFAR10-rot. The third column is trained by original training set (without rotation) and tested by CIFAR10-rot+ data set. The fourth and fifth columns are trained and tested by CIFAR10-rot and CIFAR10-rot + datasets respectively.

From Table 1 we can find that:

1) Keeping the original CNN structure and adding only the RRLs can improve the recognition accuracy of rotating images greatly;

2) Trained with the augmented data, the accuracy of improved network decreases. It implies that LeNet-5 cannot

Training Data	CIFAR-10	CIFAR-10	CIFA10-10-rot	CIFAR-10-rot+
Testing Data	CIFAR-10-rot	CIFAR-10-rot+	CIFAR-10-rot	CIFAR-10-rot+
LeNet-5	33.2	18.2	38.7	25.4
LeNet-5+*RRL*	71.3	52.8	70.9	49.1

Table 1: Comparison of accuracy (%) on LeNet-5 with and without *RRL*.

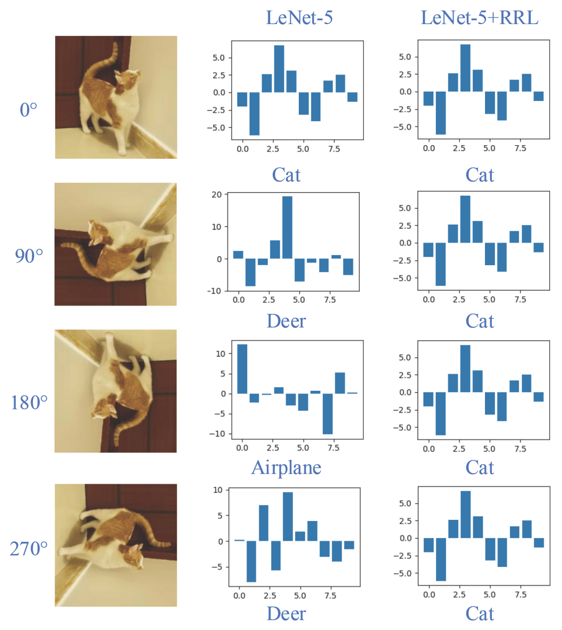

Figure 4: Feature distributions of LeNet-5 with and without *RRL*s, input with different rotation angles

.

provide more convolution kernels to learn the same patterns with different directions, so it reduces generalization;

3) Without RRLs, the accuracy of recognition can be improved a bit by using data augmentation, but the training cost increases and the problem is not solved essentially.

In order to analyze the role of *RRL* more intuitively, the feature maps are visualized in Figure 4. In Figure 4, the left columns are the input images. The middle columns are the output feature of the last layer of the original LeNet-5 network. Except the upright image can be correctly classified, the other three cases are misidentified. The right columns are the output features of the last layer of LeNet-5+RRL network, whose features do not change with the rotation angle, and all predict the correct category. It shows that with the RRLs, the same features are extracted from the images with different angles, and the coding invariance is realized.

The visualization results of Grad-CAM(Selvaraju et al. 2017) are shown in Figure 5. The input images are rotated $\theta \in \{0, 90, 180, 270\}°$. Conv1-grad, Conv2-grad and RRL3-grad are the heatmaps obtained by gradient calculation of the

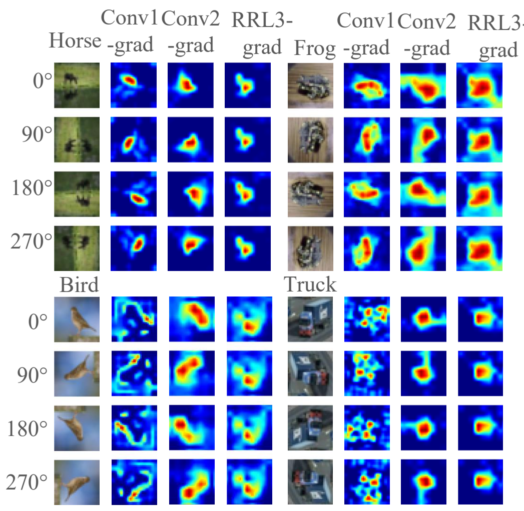

Figure 5: Visualization of three *RRL*s output heatmaps in LeNet-5 + *RRL*

Training Data	CIFAR-10	CIFAR-10	CIFA10-10-rot	CIFAR-10-rot+
Testing Data	CIFAR-10-rot	CIFAR-10-rot+	CIFAR-10-rot	CIFAR-10-rot+
ResNet-18	46.5	38.7	73.6	58.7
ResNet-18+*RRL*	75.0	65.3	77.9	63.1

Table 2: Comparison of accuracy (%) ResNet-18 with and without *RRL*.

feature maps after the first layer, the second layer of convolution and the last regional rotation layer, respectively. It can be seen that before the last regional rotation, the features are direction dependent, and the focus position of the model still changes with the rotation angle of input. At this stage, the network is only rotation equivariant. After RRL3, the neural network completes the invariance coding of rotation, and the features shown by RRL3-grad hardly change with the rotation.

Image Classification Based on ResNet-18

ResNet-18 (He et al. 2016) was proposed in 2016, and it consists of 17 convolution layers and a fully connected layer. The core component of ResNet is the residual module, which consists of two consecutive convolution layers and a skip connection.

Table 2 shows the comparison of the accuracy of ResNet-18 with and without *RRL*. It can be seen from Table 2 that the effect of data augmentation is not significant with *RRL*. No matter whether the input sample is rotated or not, as long as the sample itself remains unchanged, the local features will remain unchanged after *RRL*s.

Comparison with other methods on CIFAR-10 is shown

Training Data	CIFAR-10	
Testing Data	CIFAR-10-rot	CIFAR-10-rot+
LeNet-5	33.2	18.2
LeNet-5+*RRL*	71.3	52.8
ResNet-18	46.5	38.7
ResNet-18+*RRL*	75.0	65.3
CyResNet56-P (Cowen et al. 2015)	-	61.3
PR_RF_1 (Follmann and Bottger 2018)	-	44.1
ORN (Zhou et al. 2017)	60.9	40.7

Table 3: Comparison of accuracy (%) with other methods on CIFAR-10.

in Table 3. We can see that ResNet-18+*RRL* has obtained high accuracy on both data sets. It shows that *RRL* can help the original CNN to improve the encoding ability without increasing the parameters and model complexity, and obtain stronger generalization ability.

From table 3, we can also find that ResNet + *RRL* improves performance less than LeNet + *RRL* does (28.5% vs 38.1% on CIFAR10-rot, 26.6% vs 34.1% on CIFAR10-rot+). *LBP* operator tends to rotate the brighter texture of the image to the left part of the window. We can guess that with the restriction of *RRL*, the obtained features tend to be similar and reduce the diversity of features. After the training data are enhanced, the gap between the two is also significantly smaller (improve 4.3% on CIFAR10-rot and 4.4% on CIFAR10-rot+ respectively). For rotated images, the traditional convolutional network will specially customize the filter for each direction of the same texture. However, due to the constraint of *RRL*, even if the input data are more diverse, the feature types with little change in content will not increase significantly. Therefore, when the model is deepened, the tradition neural networks will improve more than that of networks with *RRL*.

Even with data augmentation, the rotation invariance of conventional convolution network is still not as good as plugged with *RRL*. However, it can be predicted that with the deepening of the network, the rising trend of accuracy with *RRL* will slow down. Therefore, the algorithm is suitable for shallow or medium neural networks or limited training samples and limited computing resources.

Figure 6 shows the classification results of the same image at different rotation angles with and without *RRL*s. The blue sections mean the angle range of correctly classifying ”frogs”. The image is rotated every 12°, thus there are 30 rotation angle sections. Figure 6(a) shows the classification output of the model without *RRL*s. When the rotation angle is between $\theta \in [-36, 24]$°or $\theta \in [36, 60]$°, it can be classified correctly. In other states, different prediction results will be obtained with different angles. Figure 6(b) shows the classification output of the improved model. The blue area is larger than that of (a), indicating that the rotation layers makes the model more insensitive to the input rotation angle.

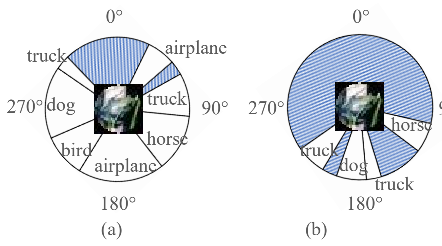

Figure 6: Recognition results with arbitrary rotation angles. (a)ResNet-18, (b)ResNet-18 + *RRL*

.

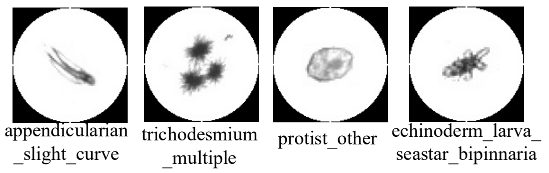
appendicularian _slight_curve
trichodesmium _multiple
protist_other
echinoderm_larva_ seastar_bipinnaria

Figure 7: Sample images in Plankton dataset

.

Plankton Recognition Based on ResNet-44

The plankton dataset(Cowen et al. 2015) consists of 30,336 gray images of different sizes, which are unevenly divided into 121 categories, corresponding to different kinds of plankton. There are 27,299 images in training set and 3,037 images in testing set. In order to unify the sample number of each category, the data are augmented for the categories with a small number of images. Finally, each category in the training set contains 2000 images and each category in the testing set contains 100 images. Each sample resizes to 50×50, then pads with white background to 64×64, and finally take its maximum inscribed circle to ensure that the image is in the centre of the image. Figure 7 shows the results of data processing and their categories.

Each image contains a single organism, which may be in any direction in three-dimensional space due to ignoring the small influence of gravity. And the ocean is full of debris particles, so there will inevitably be some noise in the image. The existence of unknown categories requires the model to deal with unrecognized objects, so it is necessary to classify those plankton with large shape differences into the same category. The above factors make the classification more difficult.

The standard ResNet-44 consists of 43 convolutional layers and a fully connected layer.A regional rotation layer is added in front of all convolutional layers.

After 100,000 epochs of training on the training set, the final plankton classification model is obtained. In order to compare the effect of regional rotation layer, the original ResNet-44 and the ResNet-44 + RRL model after adding regional rotation layer are trained and tested respectively.

Figure 8(a) shows the loss curves of the two algorithms on the training set. The orange curve is ResNet-44 + RRL

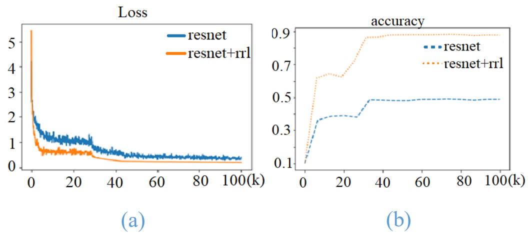

Figure 8: Comparison of loss and accuracy with and without *RRL* by ResNet-44. The blue curve is the orginal ResNet-44 model, and the orange curve is ResNet-44+*RRL*.

.

Model	Multi-class Score
ResNet-44	3.67862
ResNet-44+*RRL*	2.18777

Table 4: Comparison of loss scores of real testing sets on Plankton dataset.

model, and the blue curve is the original ResNet-44 model. It can be seen that the orange curve is always smaller than the blue curve, that is, the regional rotation layer makes the model error smaller. Figure 8(b) shows the accuracy curves of the two algorithms on the testing set. Obviously, after adding the regional rotation layer, the error of the training set is reduced, and there is no over-fitting, and the performance is improved. Table 4 shows the running results of applying the two models to the real test set without published labels. The lower the score means the model performs better. The performance of the dataset shows that without increasing the model parameters, the convolutional neural network can have stronger generalization by adding a regional rotation layer, and give the neural network the ability to capture global and local rotation.

Object Detection Based on MobileNet-tiny-yolov3

MobileNet-tiny-yolov3 is selected as the basic network. Compared with the darknet53 with residual as the main structure, mobilenet can achieve a better balance in terms of calculation, storage space and accuracy. Using the pruned tiny yolov3, the model is smaller and has more advantages when the computing resources are limited, and the fast detection speed also makes tiny yolov3 more cost-effective and easier to be applied in practice.

Rotation Transformation of Coordinate In the object detection task, the coordinates of the upper left corner and the lower right corner of the target bounding boxes are usually provided as labels, so the location changes with the rotation of the target. The corresponding coordinate labels need to be recalculated. Here we only consider four rotation angles, $\theta \in \{0, 90, 180, 270\}$ °. As shown in Figure 9. There is a rectangular box surrounding the target object, which is defined by the upper left coordinate (x_1, y_1) and the lower right coordinate (x_2, y_2). When the image ro-

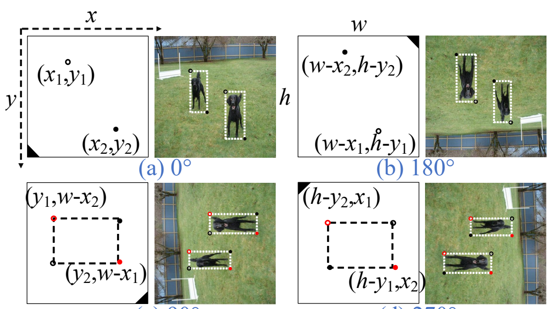

Figure 9: Coordinate transformation

.

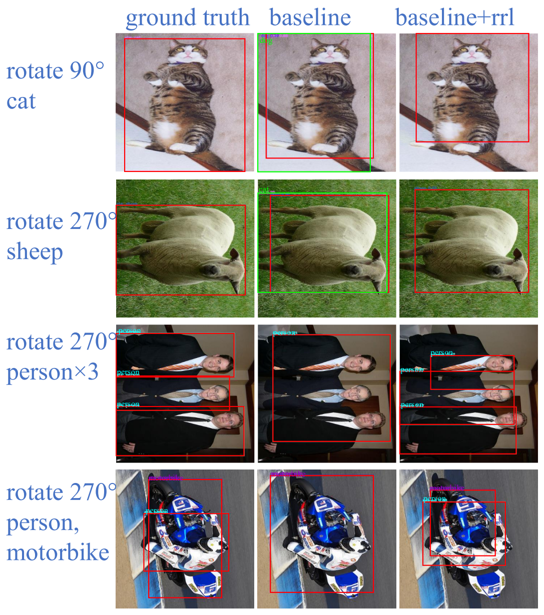

Figure 10: Examples of detection effect before and after model improvement

.

tates 90°counterclockwise, the point (x_1, y_1) is transformed into $(y_1, w - x_1)$, the point (x_2, y_2) is transformed into $(y_2, w - x_2)$, and it represents the points in the lower left corner and upper right corner of the rectangular box respectively. The final coordinate label becomes a red hollow point $(y_1, w-x_1)$ and a red solid point $(y_2, w-x_2)$. Similarly, (b) and (d) represent the position label when rotating 180°and 270°counterclockwise.

Dataset Pascal VOC dataset contains 20 categories. The dataset has been widely used in object detection, semantic segmentation and classification tasks, and as a common test benchmark. VOC 2007 and VOC 2012 are used in this experiment. Finally, 16,551 images and 40,058 objects are used in training, 4,952 images and 12,032 objects are used in testing.

Model	mAP
MobieNet-tiny-yolov3	43.76%
MobieNet-tiny-yolov3+*RRL*	61.78%

Table 5: mAP of MobileNet-tiny-yolov3 on Pascal VOC dataset. Trained by upright images and tested by rotated images.

Experimental Results and Analysis The detection effect of MobileNet-tiny-yolov3 with and without *RRL* are shown in Figure 10. Both models are trained with upright images. The first column shows the groundtruth after rotating and scaling the original image, the second and third column are the testing results on the basic model, and adding *RRL*s. For the top two rows, the image contains only one label, but the basic model outputs two prediction boxes, one of which does not belong to the correct category, as shown in the green box. It can be seen that the basic model has some recognition ability for rotating images, and will be misled into other wrong categories. The output by the improved model is quite similar with real label. The bottom two rows contain multiple labels, and they all overlap to some extent. The output of the basic model contains only one target and all objects. It shows that the model can only be roughly positioned, and can no longer be finely divided. The improved model can detect each object with some location errors. IoU threshold is set to 0.5, trained with upright pictures and tested with rotating pictures. The mAP of the two models is shown in the table 5.

Conclusion

This paper proposes a regional rotation layer (*RRL*) to help CNNs to learn rotation invariant features. By data augmentation, CNN needs to train more filters for each change in the sample, which leads to the increase of the number of parameters. So it is important to balance the network size and the data size. In this paper, *LBP* operator is used to encode the local region so that it has the same local features before and after rotation. Thus, when the input changes, the local features remain the same. Then *RRL* is integrated with LeNet-5, ResNet-18 and tiny-yolov3, which verifies the effectiveness of the method. Experimental results are analysed in detail, the applicable scenarios and shortcomings of the method are presented.

Acknowledgments

This project is supported by the fund of Science and Technology of Sichuan Province (No.2021YFG0330). Thanks for Shuyu Zhang, Yiting Wang.

References

Cheng, G.; Zhou, P.; and Han, J. 2016. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2884–2893. LAS VEGAS: IEEE.

Cohen, T.; and Welling, M. 2016. Group equivariant convolutional networks. In *International conference on machine learning*, 2990–2999. PMLR.

Cohen, T. S.; Geiger, M.; Köhler, J.; and Welling, M. 2018. Spherical CNNs. In *International Conference on Learning Representations*.

Cohen, T. S.; and Welling, M. 2017. Steerable cnns. In *International Conference on Learning Representations*.

Cowen, R. K.; Sponaugle, S.; Robinson, K.; and Luo, J. 2015. Planktonset 1.0: Plankton imagery data collected from fg walton smith in straits of florida from 2014–06-03 to 2014–06-06 and used in the 2015 national data science bowl (ncei accession 0127422). *NOAA National Centers for Environmental Information*.

Dalal, N.; and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, 886–893. Boston, MA: IEEE.

Dieleman, S.; De Fauw, J.; and Kavukcuoglu, K. 2016. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, 1889–1898. PMLR.

Dieleman, S.; Willett, K. W.; and Dambre, J. 2015. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly notices of the royal astronomical society*, 450(2): 1441–1459.

Esteves, C.; Allen-Blanchette, C.; Zhou, X.; and Daniilidis, K. 2018. Polar Transformer Networks. In *International Conference on Learning Representations*.

Follmann, P.; and Bottger, T. 2018. A rotationally-invariant convolution module by feature map back-rotation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 784–792. IEEE.

Gao, L.; Li, H.; Lu, Z.; and Lin, G. 2019. Rotation-equivariant convolutional neural network ensembles in image processing. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, 551–557. London: Association for Computing Machinery,New York.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. *Advances in neural information processing systems*, 28: 2017–2025.

Jiang, R.; and Mei, S. 2019. Polar coordinate convolutional neural network: from rotation-invariance to translation-invariance. In *2019 IEEE International Conference on Image Processing (ICIP)*, 355–359. IEEE.

Laptev, D.; Savinov, N.; Buhmann, J. M.; and Pollefeys, M. 2016. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 289–297. LAS VEGAS: IEEE.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, J.; Yang, Z.; Liu, H.; and Cai, D. 2018. Deep rotation equivariant network. *Neurocomputing*, 290: 26–33.

Maninis, K.-K.; Pont-Tuset, J.; Arbeláez, P.; and Van Gool, L. 2016. Convolutional oriented boundaries. In *European conference on computer vision*, 580–596. Amsterdam, The Netherlands: Springer.

Marcos, D.; Volpi, M.; Komodakis, N.; and Tuia, D. 2017. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 5048–5057.

Mash, R.; Borghetti, B.; and Pecarina, J. 2016. Improved aircraft recognition for aerial refueling through data augmentation in convolutional neural networks. In *International symposium on visual computing*, 113–122. Las Vegas, Nevada, USA: Springer.

Ojala, T.; Pietikäinen, M.; and Harwood, D. 1996. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1): 51–59.

Quiroga, F.; Ronchetti, F.; Lanzarini, L.; and Bariviera, A. F. 2018. Revisiting data augmentation for rotational invariance in convolutional neural networks. In *International Conference on Modelling and Simulation in Management Sciences*, 127–141. Girona, Spain: Springer.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 618–626.

Tamura, M.; Horiguchi, S.; and Murakami, T. 2019. Omnidirectional pedestrian detection by rotation invariant training. In *2019 IEEE winter conference on applications of computer vision (WACV)*, 1989–1998. Hawaii: IEEE.

Van Dyk, D. A.; and Meng, X.-L. 2001. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1): 1–50.

Wiersma, R.; Eisemann, E.; and Hildebrandt, K. 2020. Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (TOG)*, 39(4): 92–1.

Zhou, Y.; Ye, Q.; Qiu, Q.; and Jiao, J. 2017. Oriented response networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 519–528.