Oriented Response Networks

Yanzhao Zhou[1], Qixiang Ye[1], Qiang Qiu[2] and Jianbin Jiao[1]

[1]University of Chinese Academy of Sciences
[2]Duke University
zhouyanzhao215@mails.ucas.ac.cn, {qxye, jiaojb}@ucas.ac.cn, qiang.qiu@duke.edu

Abstract

Deep Convolution Neural Networks (DCNNs) are capable of learning unprecedentedly effective image representations. However, their ability in handling significant local and global image rotations remains limited. In this paper, we propose Active Rotating Filters (ARFs) that actively rotate during convolution and produce feature maps with location and orientation explicitly encoded. An ARF acts as a virtual filter bank containing the filter itself and its multiple unmaterialised rotated versions. During back-propagation, an ARF is collectively updated using errors from all its rotated versions. DCNNs using ARFs, referred to as Oriented Response Networks (ORNs), can produce within-class rotation-invariant deep features while maintaining inter-class discrimination for classification tasks. The oriented response produced by ORNs can also be used for image and object orientation estimation tasks. Over multiple state-of-the-art DCNN architectures, such as VGG, ResNet, and STN, we consistently observe that replacing regular filters with the proposed ARFs leads to significant reduction in the number of network parameters and improvement in classification performance. We report the best results on several commonly used benchmarks [1].

1. Introduction

The problem of orientation information encoding has been extensively investigated in hand-crafted features, *e.g.*, Gabor features [15, 17], HOG [9], and SIFT [31]. In Deep Convolution Neural Networks (DCNNs), the inherent properties of convolution and pooling alleviate the effect of local transitions and warps; however, lacking the capability to handle large image rotation limits DCNN's performance in many visual tasks including object boundary detection [16, 32], multi-oriented object detection [6], and image classification [20, 23].

[1]Source code is publicly available at yzhou.work/ORN

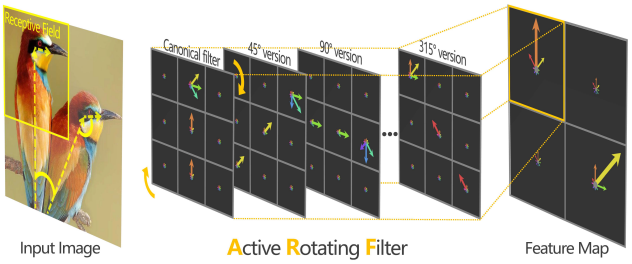


Figure 1. An ARF is a filter of the size $W \times W \times N$, and viewed as N-directional points on a $W \times W$ grid. The form of the ARF enables it to effectively define relative rotations, *e.g.*, the head rotation of a bird about its body. An ARF actively rotates during convolution; thus it acts as a virtual filter bank containing the canonical filter itself and its multiple unmaterialised rotated versions. In this example, the location and orientation of birds in different postures are captured by the ARF and explicitly encoded into a feature map.

Due to the lack of ability in fully understanding rotations, the most straightforward way for DCNN to decrease its loss is "learning by rote". The visualization of convolutional filters [11, 47] indicates that different rotated versions of one identical image structure are often redundantly learned in low-level, middle-level, and relatively high-level filters, such as those in the VGG-16 model trained on ImageNet [10]. When object parts rotate relatively to objects themselves, *e.g.*, bird's head to its body, it requires learning multiple combinations of each orientation-distinct component with more convolutional filters. In such cases, the network could give up understanding the concept of the whole object and tend to use a discriminative part of it to make the final decisions [48]. The learning-by-rote strategy needs a larger number of parameters to generate orientation-redundant filters, significantly increasing both the training time and the risk of network over-fitting. Besides, the training data is not sufficiently utilized since the limited instances are implicitly split into subsets, which could increase the possibility of filter under-fitting. To alleviate such a problem, data augmentation, *e.g.*, rotating each

training sample into multi-oriented versions, is often used. Data augmentation improves the learning performance by extending the training set. However, it usually requires more network parameters and higher training cost.

In this paper, we propose Active Rotating Filters (ARFs) and leverage Oriented Response Convolution (ORConv) to generate feature maps with orientation channels that explicitly encode the location and orientation information of discriminative patterns. Compared to conventional filters, ARFs have an extra dimension to define the arrangement of oriented structures. During the convolution, each ARF rotates and produces feature maps to capture the response of receptive fields from multiple orientations, as shown in Fig. 1. The feature maps with orientation channels carry the oriented response along with the hierarchical network to produce high-level representations, endowing DCNNs the capability of capturing global/local rotations and the generalization ability for rotated samples never seen before.

Instead of introducing extra functional modules or new network topologies, our method implements the prior knowledge of rotation to the most basic element of DCNNs, *i.e*., the convolution operator. Thus, it can be naturally fused with modern DCNN architectures, upgrading them to more expressive and compact Oriented Response Networks (ORNs). With the orientation information that ORNs produce, we can either apply SIFT-like feature alignment to achieve rotation invariance or perform image/object orientation estimation. The contributions of this paper are summarized as follows:

- We specified Active Rotating Filters and Oriented Response Convolution, improved the most fundamental module of DCNN and endowed DCNN the capability of explicitly encoding hierarchical orientation information. We further applied such orientation information to rotation-invariant image classification and object orientation estimation.
- We upgraded successful DCNNs including VGG, ResNet, TI-Pooling and STN to ORNs, achieving state-of-the-art performance with significantly fewer network parameters on popular benchmarks.

2. Related Works

2.1. Hand-crafted features.

Orientation information has been explicitly encoded in classical hand-crafted features including Weber's Law descriptor [5], Gabor features [15, 17], SIFT [31], and LBP [33, 1]. SIFT descriptor [31] and its modification with affine-local regions [25] find the dominant orientation of a feature point, according to which statistics of local gradient directions of image intensities are accumulated to give a summarizing description of local image structures. With dominant orientation based feature alignment, SIFT achieves invariance to rotation and robustness to moderate perspective transforms [2, 12]. Starting from the gray values of a circularly symmetric neighbor set of pixels in a local neighborhood, LBP derives an operator that is by definition invariant against any monotonic transformation of the gray scale [33, 1]. Rotation invariance is achieved by minimizing the LBP code value using the bit cyclic shift. Other representative descriptors including CF-HOG [39] that uses orientation alignment and RI-HOG [30] that leverages radial gradient transform to be rotation invariant.

2.2. Deep Convolutional Neural Networks.

Deep Convolution Neural Networks have the capability of processing transforms including moderate transitions, scale changes, and small rotations. Such capability is endowed with the inherent properties of convolutional operations, redundant convolutional filters, and hierarchical spatial pooling [35, 20]. More general pooling operations [26] permit to consider invariance to local deformation that however does not correspond to specific prior knowledge.

Data augmentation. Given rich, and often redundant, convolutional filters, data augmentation can be used to achieve local/global transform invariance [43]. Despite the effectiveness of data augmentation, the main drawback lies in that learning all the possible transformations of augmented data usually requires more network parameters, which significantly increases the training cost and the risk of over-fitting. Most recent TI-Pooling [23] alleviates the drawbacks by using parallel network architectures for the considered transform set and applying the transform invariant pooling operator on their outputs before the top layer. The essence of TI-Pooling comprises multi-instance learning and weight sharing which help to find the most optimal canonical instance of the input images for training, as well as reducing the redundancy in learned networks. Nevertheless, with built-in data augmentation, TI-Pooling requires significantly more training and testing cost than a standard DCNN.

Spatial Transform Network. Representatively, the spatial transformer network (STN) [20] introduces an additional network module that can manipulate the feature maps according to the transform matrix estimated with a localisation sub-CNN. STN contributes a general framework for spatial transform, but the problem about how to precisely estimate the complex transform parameters by CNN remains not being well-solved [14, 34]. In [21, 36], the Convolutional Restricted Boltzmann Machine (C-RBM) induces transformation-aware filters, *i.e*., it yields filters that have a notion with which specific image transformation they are used. From the view of group theory, Cohen *et al*. [8] justified that the spatial transform of images could be reflected in both feature maps and filters, providing a theoretical foundation for our work. Most recent works

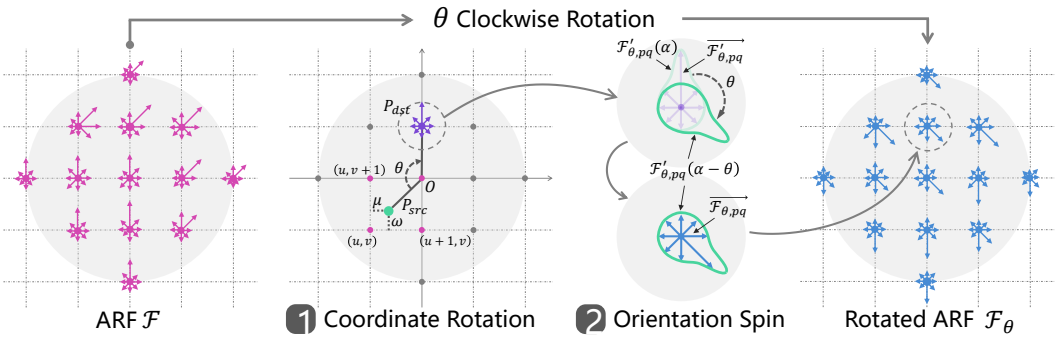


Figure 2. An ARF $\mathcal{F}$ is clockwise rotated by θ to yield its rotated variant $\mathcal{F}_\theta$ in two steps: coordinate rotation and orientation spin.

[44, 13] have tried rotating conventional filters to perform rotation-invariant texture and image classification; however, without upgrading conventional filters to multi-oriented filters with orientation channels, their capability about capturing hierarchical and fine-detailed orientation information remains limited.

3. Oriented Response Networks

Oriented Response Networks (ORNs) are deep convolutional neural networks using Active Rotating Filters (ARFs). An ARF is a filter that actively rotates during convolution to produce a feature map with multiple orientation channels. Thus, an ARF acts as a virtual filter bank with only one filter being materialized and learned. With ARFs, ORNs require significantly fewer network parameters with negligible computation overhead and enable explicitly hierarchical orientation information encoding.

In what follows, we address three problems in adopting ARFs in DCNN. First, we construct a two-step technique to efficiently rotate an ARF based on the circular shift property of Fourier Transform. Second, we describe convolutions that use ARFs to produce feature maps with location and orientation explicitly encoded. Third, we show how all rotated versions of an ARF contribute to its learning during the back-propagation update stage.

3.1. Active Rotating Filters

An Active Rotating Filter (ARF) is a filter of the size $W \times W \times N$ that actively rotates $N-1$ times during convolution to produce a feature map of N orientation channels, Fig. 2. Therefore, an ARF $\mathcal{F}$ can be virtually viewed as a bank of N filters ($N \times W \times W \times N$), where only the canonical filter $\mathcal{F}$ itself is materialized and to be learned, and the remaining $N-1$ filters are its unmaterialized copies. The n-th filter in such a filter bank, $n \in [1, N-1]$, is obtained by clockwise rotating $\mathcal{F}$ by $\frac{2\pi n}{N}$.

An ARF contains N orientation channels and is viewed as N-directional points on a $W \times W$ grid. Each element in an ARF $\mathcal{F}$ can be accessed with $\overrightarrow{\mathcal{F}_{ij}}^{(n)}$ where $0 \leq |i|, |j| \leq \frac{W-1}{2}, 0 \leq n \leq N-1, i, j, n \in \mathbb{N}$. An ARF $\mathcal{F}$ is clockwise rotated by θ to yield its rotated variant $\mathcal{F}_\theta$ through the following two steps, coordinate rotation and orientation spin.

Coordinate Rotation. An ARF rotates around the origin O, Fig. 2, and the point at (p, q) in $\mathcal{F}_\theta$ is calculated from four neighbors around (p', q') in $\mathcal{F}$, $\begin{pmatrix} p' & q' \end{pmatrix} = \begin{pmatrix} p & q \end{pmatrix} \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix}$, using bilinear interpolation

$$\begin{aligned} \overrightarrow{\mathcal{F'}_{\theta,pq}} = (1-\mu)(1-\omega)\overrightarrow{\mathcal{F}_{uv}} + (1-\mu)\omega\overrightarrow{\mathcal{F}_{u,v+1}} \\ + \mu(1-\omega)\overrightarrow{\mathcal{F}_{u+1,v}} + \mu\omega\overrightarrow{\mathcal{F}_{u+1,v+1}}, \end{aligned} \tag{1}$$

where $u = \lfloor p' \rfloor, v = \lfloor q' \rfloor, \mu = p' - u, \omega = q' - v$. Note that points outside the inscribed circle are padded with 0.

Orientation Spin. As discussed, an ARF can be viewed as N-directional points on a grid. Each N-directional point $\overrightarrow{\mathcal{F'}_{\theta,pq}}$ is the N-points uniform sampling of a desired oriented response $\mathcal{F'}_{\theta,pq}(\alpha)$, which is a continuous periodic function of angle α with period 2π. After the coordinates rotation, it still requires a clockwise spin by θ to yield $\overrightarrow{\mathcal{F}_{\theta,pq}}$, which is, in fact, the quantization of $\mathcal{F'}_{\theta,pq}(\alpha - \theta)$, Fig. 2. Therefore, such spin procedure can be efficiently tackled in Fourier domain by using the circular shift property of Discrete Fourier Transforms (DFT),

$$\begin{aligned} X(k) &\equiv \mathbf{DFT}\{\overrightarrow{\mathcal{F'}_{\theta,pq}}^{(n)}\} \\ &= \sum_{n=0}^{N-1} \overrightarrow{\mathcal{F'}_{\theta,pq}}^{(n)} e^{-jk\frac{2\pi n}{N}}, k=0,1,\ldots,N-1, \end{aligned} \tag{2}$$

$$\begin{aligned} \overrightarrow{\mathcal{F}_{\theta,pq}}^{(n)} &\equiv \mathbf{IDFT}\{X(k)e^{-jk\theta}\} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{jk(\frac{2\pi n}{N}-\theta)}, n=0,1,\ldots,N-1. \end{aligned} \tag{3}$$

To smoothly process all rotation angles, ARFs require a considerable amount of orientation channels. In practice, thanks to the orientation 'interpolation' by multi-layer pooling operations, we can use a limited amount of orientations

to guarantee the accuracy. The successful practice of DCNNs, *e.g.*, VGG [38] and ResNet [18, 19], shows that the stacks of multiple small filters are more expressive and parameters-efficient than large filters. Moreover, when using the combination of small filters and a limited number of orientation channels, the computational complexity of rotating ARF can be further reduced, since both the coordinate rotation and the orientation spin can be calculated by the circular shift operator and implemented via high-efficient memory mapping under reasonable approximations. Take a $3 \times 3 \times 8$ ARF $\hat{\mathcal{F}}$ as an example, calculations of its θ clockwise rotated version $\hat{\mathcal{F}}_\theta$ are formulated as

$$\begin{aligned} \overrightarrow{\hat{\mathcal{F}}'_{\theta,\langle i\rangle}} &= \overrightarrow{\hat{\mathcal{F}}'_{\langle (i-k) \bmod N\rangle}}, i\in\mathcal{I}, \\ \overrightarrow{\hat{\mathcal{F}}_\theta^{(n)}} &= \overrightarrow{\hat{\mathcal{F}}'^{((n-k) \bmod N)}_\theta}, n=0,1,\ldots,N-1, \end{aligned} \tag{4}$$

where $\forall k \in \mathbb{N}, \theta = k\frac{2\pi}{N}, N = 8$ and $\mathcal{I} = \begin{pmatrix} 7 & 0 & 1 \\ 6 & & 2 \\ 5 & 4 & 3 \end{pmatrix}$ is a mapping table that defines the index of each surrounding element, which means $\overrightarrow{\hat{\mathcal{F}}_{\langle 0\rangle}} \equiv \overrightarrow{\hat{\mathcal{F}}_{0,1}}$, $\overrightarrow{\hat{\mathcal{F}}_{\langle 1\rangle}} \equiv \overrightarrow{\hat{\mathcal{F}}_{1,1}}$, $\overrightarrow{\hat{\mathcal{F}}_{\langle 2\rangle}} \equiv \overrightarrow{\hat{\mathcal{F}}_{1,0}}$, $\overrightarrow{\hat{\mathcal{F}}_{\langle 3\rangle}} \equiv \overrightarrow{\hat{\mathcal{F}}_{1,-1}}$ and so on.

Given the above, we use 1×1 and 3×3 ARFs with 4 and 8 orientation channels in most experiments.

3.2. Oriented Response Convolution

An ARF actively rotates $N-1$ times during convolution to produce a feature map of N orientation channels, and such feature map explicitly encodes both location and orientation information. As an ARF is defined as the size $W \times W \times N$, both an ARF $\mathcal{F}$ and an N-channel feature map $\mathcal{M}$ can be viewed as N-directional points on a grid. With ARF, we define the Oriented Response Convolution over $\mathcal{F}$ and $\mathcal{M}$, denoted as $\tilde{\mathcal{M}} = \mathbf{ORConv}(\mathcal{F}, \mathcal{M})$. The output feature map $\tilde{\mathcal{M}}$ consists of N orientation channels and the k-th channel is computed as

$$\tilde{\mathcal{M}}^{(k)} = \sum_{n=0}^{N-1} \mathcal{F}_{\theta_k}^{(n)} * \mathcal{M}^{(n)}, \theta_k = k\frac{2\pi}{N}, k=0,\ldots,N-1, \tag{5}$$

where $\mathcal{F}_{\theta_k}$ is the clockwise θ_k-rotated version of $\mathcal{F}$, $\mathcal{F}_{\theta_k}^{(n)}$ and $\mathcal{M}^{(n)}$ are the n-th orientation channel of $\mathcal{F}_{\theta_k}$ and $\mathcal{M}$ respectively.

According to (5), the k-th orientation channel of the output feature map $\tilde{\mathcal{M}}$ is generated by θ_k rotated versions of the materialised ARF. It means that in each oriented response convolution, the ARF proactively captures image response in multiple directions and explicitly encodes its location and orientation into a single feature map with multiple orientation channels, visualized in Fig. 3. (5) also demonstrates that each orientation channel of the ARF contributes to the final convolutional response respectively, endowing ORNs the capability of capturing richer and more fine-detailed patterns than a regular CNN.

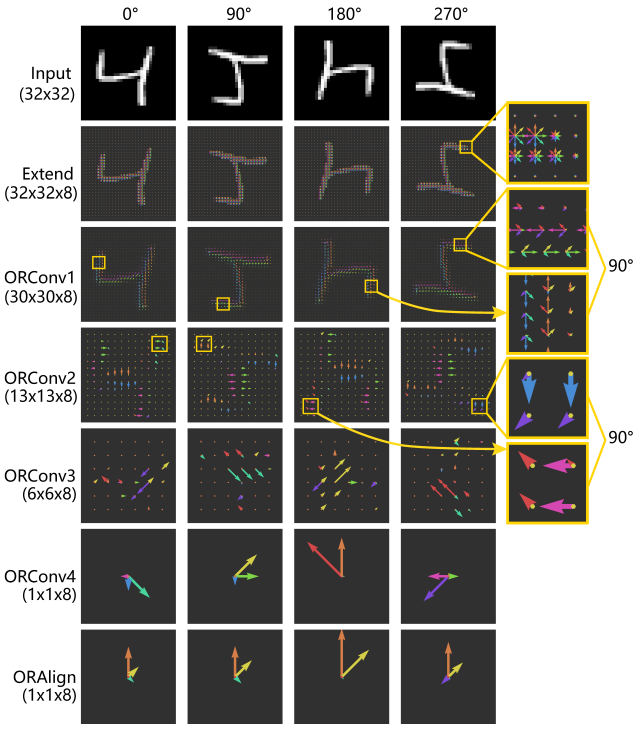


Figure 3. Example feature maps produced by one ARF at each layer of an ORN trained on the rotated MNIST dataset, with digit '4' in different rotations as the inputs (one network layer per row, one input per column). The right-most column magnifies sample regions in feature maps. It clearly shows that a feature map explicitly encodes position and orientation. At the second layer, an image is extended to an omnidirectional map to fit ORConv. At the second-to-last (ORConv4) layer, deep features are observed in similar values but in different orientations, which demonstrates that orientation information is extracted by ORNs. The last (ORAlign) layer performs SIFT-like alignment to enable rotation-invariance (Best viewed zooming on screen).

3.3. Updating Filters

During the back-propagation, error signals $\delta^{(k)}$ of all rotated versions of the ARF are aligned to $\delta^{(k)}_{-\theta_k}$ using (1) and (2), and aggregated to update the materialised ARF,

$$\begin{aligned} \delta^{(k)} &= \frac{\partial L}{\partial \mathcal{F}_{\theta_k}}, \theta_k = k\frac{2\pi}{N}, k=0,1,\ldots,N-1, \\ \mathcal{F} &\leftarrow \mathcal{F} - \eta \sum_{0}^{N-1} \delta^{(k)}_{-\theta_k}, \end{aligned} \tag{6}$$

where L stands for training loss and η for learning rate. An ARF acts as a virtual filter bank containing the materialized canonical filter itself and unmaterialised rotated versions. According to (6), the back-propagation collectively updates the materialised filter only, so that training errors of appearance-like but orientation-distinct samples are aggregated. In low-level layers, such collective updating

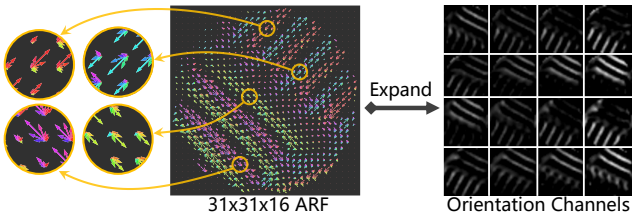


Figure 4. A $31 \times 31 \times 16$ ARF learned from a texture dataset. It is shown in the N-directional points form (left) and further visualized as one orientation channel per image (right). The ARF clearly defines a texture pattern through a combination of multi-oriented edges (Best viewed zooming on screen).

contributes more significantly, as in a single image there exist many appearance-like but orientation-distinct patches that can be exploited. The collective updating also helps when only limited training samples are given. One example of a collectively updated ARF is shown in Fig. 4.

3.4. Rotation Invariant Feature Encoding

Feature maps in ORNs are not rotation-invariant as orientation information are encoded instead of being discarded. When within-class rotation-invariance is required, we introduce two strategies, ORAlign and ORPooling, at the top layer of ORNs. For simplicity, we choose a DCNN architecture, where the size of a feature map gradually shrinks to $1 \times 1 \times N$. N is the number of orientation channels. Each feature map of the last ORConv layer has a receptive field of image size and stands for the oriented response of high-level representative patterns.

The first strategy is the ORAlign. Without loss of generality, let us denote the i-th feature map of the last ORConv layer as $\overrightarrow{\hat{\mathcal{M}}\{i\}}$ and each oriented response in it as $\overrightarrow{\hat{\mathcal{M}}\{i\}}^{(n)}, 0 \leq n \leq N-1$. $\overrightarrow{\hat{\mathcal{M}}\{i\}}$ is an N dimension tensor records the response from different directions, with which we perform SIFT-like alignment to achieve rotation robustness. This is done by first calculating the dominant orientation (the orientation with the strongest response) as $D = \underset{d}{\operatorname{argmax}} \overrightarrow{\hat{\mathcal{M}}\{i\}}^{(d)}$ and spin the feature by $-D\frac{2\pi}{N}$, Fig. 3. The second strategy is the ORPooling, which is done via simply pooling a $\overrightarrow{\hat{\mathcal{M}}\{i\}}$ to a scalar $\max(\overrightarrow{\hat{\mathcal{M}}\{i\}}^{(j)}), 0 < j < N-1$. This strategy reduces the feature dimension but loses feature arrangement information.

4. Experiments

ORNs are evaluated on three benchmarks. In Sec. 4.1, experiments on the MNIST dataset [29] and its $[0, 2\pi]$ randomly rotated versions are conducted, showing the advantage of ORNs through encoding rotation-invariant features, and reducing network parameters. ORNs are further tested on a small sample set of $[0, 2\pi]$ rotated MNIST

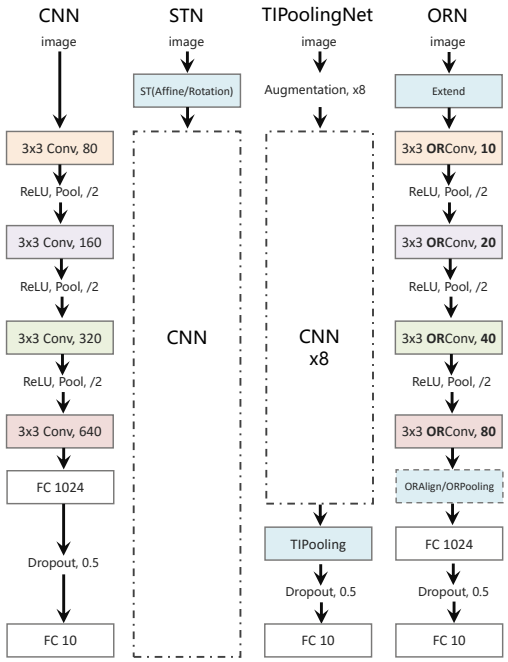


Figure 5. Comparison of network topologies.

[24] to validate its generalization ability on rotation. In Sec. 4.2, on a weakly-supervised orientation estimate task, the vast potential of directly taking advantage of the orientation information extracted by ORNs is demonstrated. In Sec. 4.3, we upgrade the VGG [38], ResNet [18], and the WideResNet [45] to ORNs, and train them on CIFAR10 and CIFAR100 [22], showing the state-of-the-art performance on the natural image classification task.

4.1. Rotation Invariance

Rotated MNIST. We randomly rotate each sample in the MNIST dataset [29] between $[0, 2\pi]$ to yield *MNIST-rot*. To assess the effect of data augmentation on different models, we further rotate each sample in the *MNIST-rot* training set to eight directions with 45-degree intervals, which means that the training set is augmented eightfold. The augmented data set is identified as *MNIST-rot+*.

We set up a baseline CNN with four convolutional layers and multiple 3x3 filters, Fig. 5. With the baseline CNN, we generate different ORNs, as well as configuring the STNs [20] and the TI-Pooling network [23] for comparison. STNs are created by inserting a Spatial Transformer with affine or rotation transform to the entry of the baseline CNN. TIPooling network is constructed by duplicating the baseline CNN eight times to capture different augmented rotated versions of inputs, and a transform-invariant pooling layer before the output layer. ORNs are built by upgrading each convolution layer in the baseline CNN to Oriented Response Convolution layer using Active Rotating Filters (ARFs) with 4 or 8 orientation channels. Considering that

Method	time(s)	params(%)	original(%)	rot(%)	rot+(%)	original → rot(%)
Baseline CNN	16.4	100.00	0.73	2.82	2.19	56.28
STN(affine)[20]	18.5	100.40	0.61	2.52	1.82	56.44
STN(rotation)[20]	18.7	100.39	0.66	2.88	1.93	55.59
TIPooling(x8)[23]	126.7	100.00	0.97†	not permitted	1.26	not permitted
ORN-4(None)	7.9	15.91	0.63	1.88	1.55	59.67
ORN-4(ORPooling)	8	7.95	0.59	1.84	1.33	27.74
ORN-4(ORAlign)	8.1	15.91	**0.57**	1.69	1.34	27.92
ORN-8(None)	17.5	31.41	0.79	1.57	1.33	58.98
ORN-8(ORPooling)	17.9	12.87	0.66	**1.37**	1.21	16.67
ORN-8(ORAlign)	17.8	31.41	0.59	1.42	**1.12**	**16.21**

Table 1. Results on the MNIST variants. The second column describes the average training time of an epoch on the *original* training set (with a NVIDIA Tesla K80 GPU). The third column describes the percentage of parameters of each model about the baseline CNN. The fourth to sixth columns describe the error rates on the *original*, the *rot*, and the *rot+* datasets. The last column describes the error rates achieved on the *rot* testing set (with random rotation) by models trained on the *original* training set (without rotation). TIPooling requires augmented data; thus some experiments are not permitted. The error rate of TIPooling on the original MNIST dataset is under augmentation, with the superscript † to show its difference with others.

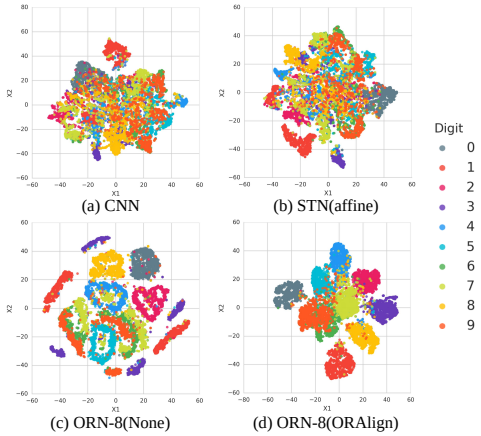


Figure 6. Visualization of features in cross-generalization evaluation, corresponding to the last column of Tab. 1.

ARFs are more expressive than conventional filters, the number of ARFs in each layer is decreased to one-eighth of those in the baseline. Corresponding to the strategies proposed in Sec. 3.4, we use ORAlign, ORPooling or none of them to encode rotation-invariant features. The network topologies are shown in Fig. 5.

In network training, we use the same hyper-parameters as TI-Pooling [23], *i.e.*, 200 training epochs using the turning-free convergent adadelta algorithm [46], 128 batch size, and 0.5 dropout rate for the fully-connected layer. For each dataset, we randomly selected 10,000 samples from the training set for validation and the remaining 50,000 samples for training. The best model selected by 5-fold cross-validation is then applied to the test set, and the final results are presented in Tab. 1.

The second column of Tab. 1 shows that ORN keeps high training efficiency. The ORN-4 (4 orientation channels) uses only 50% training time while ORN-8 uses similar training time with the baseline CNN. In contrast, TIPooling increases the time by about eight times as each sample is augmented to 8 orientations. From the third to the last column of Tab. 1, it can be seen that ORNs can use significantly fewer network parameters (7.95%-31.4%) to consistently improve the performance. Even on the original dataset without sample rotations, it achieves 22% error rate decrease (0.57% vs 0.73%), as the digit curvatures are well modeled by ORN. Compared with the data augmentation strategy (baseline CNN on *rot+*), ORN (on *rot*) not only reduces network parameters and training cost but also achieves significant lower error rate (1.37% vs 2.19%).

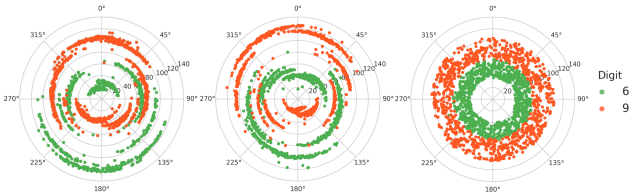
Tab. 1 also shows that different rotation-invariant encoding strategies have different advantages. ORPooling can further compress the feature dimension and network parameters, while ORAlign retains the complete feature structure thus achieves higher performance. Even without rotation-invariant encoding, ORNs outperforms the baseline on the *rot* and *rot+*, because ARFs can explicitly capture the response in different directions so that a pattern and its rotated versions can be encoded in the same feature map with orientation channels, Fig. 3. It also can be seen in Fig. 6(c) that the t-SNE [42] 2D mapping of features produced by ORN-8(None) constitutes clear clusters.

In Tab. 1, the state-of-the-art spatial transform network, STN, has minor improvement on the *rot* while slightly increasing the number of parameters. The visualization of calibrated images shows that it often outputs wrong transform parameters. This validates our previous viewpoint: the conventional CNN used in STN lacks the capability to precisely estimate rotation parameters. In Sec. 4.2, we will show that ORN can better solve such a problem.

The last column of Tab. 1 presents the results of cross-generalization evaluation that trains models on the *MNIST-original* and tests them on the *MNIST-rot*. ORNs show impressing performance with 71% improvement over the

Method	Error(%)
ScatNet-2 [3]	7.48
PCANet-2 [4]	7.37
TIRBM [40]	4.2
CNN	4.34
ORN-8(ORAlign)	**2.25**
TIPooling(with augmentation) [23]	1.93
OR-TIPooling(with augmentation)	**1.54**

Table 2. Classification error rates on the *MNIST-rot-12k*.



(a) CNN (b) STN(affine) (c) ORN-8(ORAlign)

Figure 7. Visualization of features encoding of digit class '6' and '9' from *MNIST-rot*. Each point (r, θ) corresponds to a sample where radius r is the 1-D tSNE feature mapping, and θ is the angle of the sample. ORN-8(ORAlign) produces within-class rotation-invariant deep features while maintaining inter-class discrimination. (Best viewed in color.)

baseline. Fig. 6(d) shows that ORN-8(ORAlign) produces much clearer feature distribution in manifold than other networks.

An interesting experiment comes from the digit class '6' and '9'. It can be seen in Fig. 7 that both CNN and STN have large within-class differences as the same digit with different angles produce various radii. Moreover, features generated by CNN and STN have apparently 180^o symmetrical distribution, which means that they can barely tell the difference between upside-down 6 and 9. In contrast, ORN-8(ORAlign) generates within-class rotation-invariant deep features, while maintaining inter-class discrimination.

Rotated Small Sample Set. A smaller dataset can better test the generalization capability of a learning model. We consider the *MNIST-rot-12k* dataset [24] which contains 12,000 training samples and 50,000 test samples from the *MNIST-rot* dataset. Among them, 2000 training samples are used as the validation set and the remaining 10,000 samples as the training set.

In the dataset, we test the ORN-8 model that uses 8-orientation ARFs and an ORAlign operator. We also test the OR-TIPooling network, which is constructed by upgrading its parallel CNNs to ORN-8(None)s. The reason why we do not use ORAlign or ORPooling is that TIPooling itself has the invariant encoding operator. Tab. 2 shows that ORN can decrease the state-of-the-art error rate from 4.2% to 2.25% using only 31% network parameters of the baseline CNN. Combined with TIPooling, ORN further decreases the error rate to 1.54%, achieving state-of-the-art performance,

Method	Std	Error(%)
STN [20]	0.745	3.38
OR-STN(ORAlign)	0.749	3.61
OR-STN	**0.397**	**2.43**

Table 3. Orientation estimation performance. The second column describes the standard deviation of calibrated orientations and the third column describes the classification error rates.

(a) Input (b) STN(affine) (c) OR-STN(ORAlign) (d) OR-STN

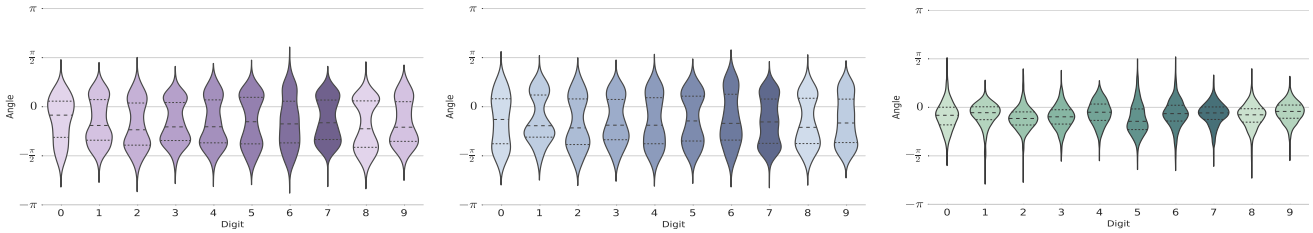
Figure 8. Orientation estimation. (a) is a mini-batch of samples from *MNIST-half-rot* and (b)-(d) are their rotation-rectified results.

which shows that ORNs have good generalization capability for such reduced training sample cases.

4.2. Orientation Estimation

ORN is evaluated on the weakly image orientation estimation problem, using the STN [20] as the baseline. The training images have only class labels but lack orientation annotation, which is estimated during learning. We upgrade the localisation sub-network of STN from a conventional CNN to ORN by converting Conv layers to ORConv layers which use ARFs with eight orientation channels. The STN model is simplified to process rotation only, which means that its localisation network estimates only a rotation parameter.

STN, OR-STN and OR-STN(ORAlign) are trained on the *MNIST-half-rot* dataset which is built by randomly rotating each sample in the MNIST dataset in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ (half the circle). All the networks use hyper-parameters as Sec. 4.1 and are trained by only 80 epochs to make the localisation sub-network converge. The orientation estimation results are presented in Tab. 3, the rotation-rectified images are shown in Fig. 8, and angle statistics of rotation-rectified images are shown in Fig. 9. It can be seen in Fig. 8(b) that STN cannot effectively handle the large-angle rotation problem, because the localisation sub-network itself is a conventional CNN, lacking the ability to explicitly process significant rotation. When upgrading the localisation network of STN to ORN (without ORAlign), it can be seen in Fig. 8(d) that most digit orientations are correctly estimated. In Fig. 9(b), it can be seen that the OR-STN(ORAlign) performs even worse than the baseline on orientation estimation, because after the feature alignment, features become rotation-invariant and thus lose orientation information. Tab. 3 shows that upgrading localisation sub-network to ORN significantly improves the performance. Such experiments validate that the ARFs can capture the orientation information of



(a) STN (b) OR-STN(ORAlign) (c) OR-STN

Figure 9. Distributions of samples' orientations after rotation-rectification.

Method	depth-k	params	CIFAR10(%)	CIFAR100(%)
NIN [28]	-	-	8.81	35.67
DSN [27]	-	-	8.22	34.57
Highway [41]	-	-	7.72	32.39
ELU [7]	-	-	6.55	24.28
VGG [38]	16	20.1M	6.32	28.49
OR-VGG	16-$\frac{1}{2}$	10.1M	5.47	27.03
ResNet [18]	110	1.7M	6.43	25.16
OR-ResNet	110-$\frac{1}{2}$	0.9M	5.31	-
pre-act-ResNet[19]	110	1.1M	6.37	-
	164	1.7M	5.46	24.33
	1001	10.3M	4.92	22.71
WideResNet[45]	40-4	8.9M	4.53	21.18
	16-8	11.0M	4.27	20.43
	28-10	36.5M	3.89	18.85
OR-WideResNet	40-$\frac{1}{2}$	1.1M	4.34	23.19
	40-2	4.5M	3.43	18.82
	28-5	18.2M	**2.98**	**16.15**

Table 4. Results on the natural image classification benchmark. In the second column, k is the widening factor corresponding to the number of filters in each layer.

discriminative patterns and explicitly encode them into feature maps with orientation channels, which are effective for image orientation estimation.

4.3. Natural Image Classification

Although most objects in natural scene images are upright, rotations could exist in small and/or medium scales (from edges to object parts). It is interesting to validate whether ORNs are effective to handle such partial object rotation or not. CIFAR-10 and CIFAR-100 datasets [22] consist of 32x32 real-world object images drawn from 10 and 100 classes split into 50,000 training and 10,000 testing images. Three DCNNs including VGG [38], ResNet [18] and WideResNet [45], are used as baselines on these datasets. Promoting the baselines to ORNs is done by converting each Conv layer to an ORConv layer that uses ARFs with eight orientation channels, and using an additional ORAlign layer to encode rotation invariant representations.

Following the **V2** settings of WideResNet [45], image classification results, Tab. 4, show that ORNs consistently improved baselines with much fewer parameters. For example, OR-VGG uses about 50% parameters of the baseline to achieve better results. OR-WideResNet-40-2 (without dropout) uses only 12% parameters (4.5M vs 36.5M) to outperform the state-of-the-art WideResNet-28-10 (with dropout) on CIFAR10. OR-WideResNet-28-5 (with dropout) uses about 50% parameters of the baselines yet significantly improve the state-of-the-arts on both CIFAR10 and CIFAR100. The top-3 improved classes of CIFAR10 are **frog** (31% higher than baseline ResNet), **bird** (30.7%) and **deer** (27.3%), which happen to involve significant local and/or global rotations, Fig. 10. This further demonstrates the capability of ORN to process local and global image rotations.

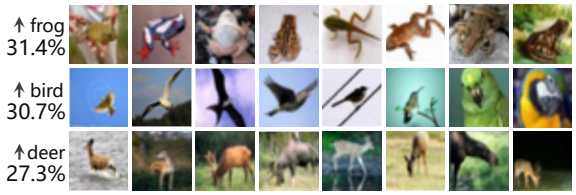


Figure 10. Sample images that contain rotated objects/parts falsely classified by the ResNet but correctly recognized by the proposed ORNs in CIFAR10.

5. Conclusions

In this paper, we proposed a simple but effective strategy to explicitly encode hierarchical orientation information of discriminative patterns and handle the global/local rotation problem. The primary contribution is designing Active Rotating Filters (ARFs), as well as upgrading the state-of-the-art DCNN architectures, *e.g.*, VGG, ResNet, STN, and TI-Pooling, to Oriented Response Networks (ORNs). Experimentally, ORNs outperform the baseline DCNNs while using significantly fewer (12%-50%) network parameters, which indicates that the usage of model-level rotation prior is a key factor in training compact and effective deep networks.

Acknowledgements

The authors are very grateful for support by NSFC grant 61671427, BMSTC grant Z161100001616005, STIFCAS grant CXJJ-16Q218, and NSF.

References

[1] T. Ahonen, A. Hadid, and M. Pietikäinen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(12):2037–2041, 2006. 2

[2] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. On the use of SIFT features for face authentication. In *CVPR Workshops*, page 35, 2006. 2

[3] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, 2013. 7

[4] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE Trans. Image Processing*, 24(12):5017–5032, 2015. 7

[5] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikäinen, X. Chen, and W. Gao. WLD: A robust local image descriptor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1705–1720, 2010. 2

[6] G. Cheng, P. Zhou, and J. Han. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *CVPR*, June 2016. 1

[7] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. 8

[8] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, pages 2990–2999, 2016. 2

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 1

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1

[11] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009. 1

[12] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, pages 1–8, 2007. 2

[13] D. M. Gonzalez, M. Volpi, and D. Tuia. Learning rotation invariant convolutional filters for texture classification. *CoRR*, abs/1604.06720, 2016. 3

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2

[15] G. M. Haley and B. S. Manjunath. Rotation-invariant texture classification using modified gabor filters. In *ICIP*, pages 262–265, 1995. 1, 2

[16] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *CVPR*, pages 1732–1740, 2015. 1

[17] J. Han and K. Ma. Rotation-invariant and scale-invariant gabor features for texture image retrieval. *Image Vision Comput.*, 25(9):1474–1481, 2007. 1, 2

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4, 5, 8

[19] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016. 4, 8

[20] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015. 1, 2, 5, 6, 7

[21] J. J. Kivinen and C. K. I. Williams. Transformation equivariant boltzmann machines. In *ICANN*, pages 1–9, 2011. 2

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009. 5, 8

[23] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. *CoRR*, abs/1604.06318, 2016. 1, 2, 5, 6, 7

[24] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007. 5, 7

[25] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. In *BMVC*, pages 1–10, 2004. 2

[26] C. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, pages 464–472, 2016. 2

[27] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 8

[28] M. Lin, Q. Chen, and S. Yan. Network In Network. *ICLR*, 2014. 8

[29] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003. 5

[30] K. Liu, H. Skibbe, T. Schmidt, T. Blein, K. Palme, T. Brox, and O. Ronneberger. Rotation-invariant HOG descriptors using fourier analysis in polar and spherical coordinates. *International Journal of Computer Vision*, 106(3):342–364, 2014. 2

[31] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999. 1, 2

[32] K. Maninis, J. Pont-Tuset, P. A. Arbeláez, and L. J. V. Gool. Convolutional oriented boundaries. In *ECCV*, pages 580–596, 2016. 1

[33] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002. 2

[34] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. 2

[35] D. Scherer, A. C. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN*, pages 92–101, 2010. 2

[36] U. Schmidt and S. Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *CVPR*, pages 2050–2057, 2012. 2

[37] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *CVPR*, pages 1233–1240, 2013.

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4, 5, 8

[39] H. Skibbe and M. Reisert. Circular fourier-hog features for rotation invariant object detection in biomedical images. In *ISBI*, pages 450–453, 2012. 2

[40] K. Sohn and H. Lee. Learning invariant representations with local transformations. In *ICML*, 2012. 7

[41] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. 8

[42] L. Van Der Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[43] D. a. van Dyk and X.-L. Meng. The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001. 2

[44] F. Wu, P. Hu, and D. Kong. Flip-rotate-pooling convolution and split dropout on convolution neural networks for image classification. *CoRR*, abs/1507.08754, 2015. 3

[45] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 5, 8

[46] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. 6

[47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014. 1

[48] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016. 1