RED-NN: Rotation-Equivariant Deep Neural Network for Classification and Prediction of Rotation

Rosemberg Rodriguez Salas, Petr Dokládal, Eva Dokladalova

Rotation-Equivariant Deep Neural Network for Classification and Prediction of Rotation

Rosemberg Rodriguez Salas[1]
r.rodriguez@esiee.fr
Petr Dokladal[2]
petr.dokladal@mines-paristech.fr
Eva Dokladalova[1]
eva.dokladalova@esiee.fr

[1] University Paris-Est
LIGM UMR 8049
Noisy le Grand, France

[2] MINES ParisTech
PSL Research University
CMM, Center for Mathematical Morphology
Fontainebleau, France

Abstract

In this work, we propose a new Convolutional Neural Network (CNN) for classification of rotated objects. This architecture is built around an ordered ensemble of oriented edge detectors to create a roto-translational space that transforms the input rotation into translation. This space allows the subsequent predictor to learn the internal spatial and angular relations of the objects regardless of their orientation. No data augmentation is needed and the model remains significantly smaller. It presents a self-organization capability and learns to predict the class and the rotation angle without requiring an angle-labeled dataset.

We present the results of training with both up-right and randomly rotated datasets. The accuracy outperforms the current state of the art on up-right oriented training.

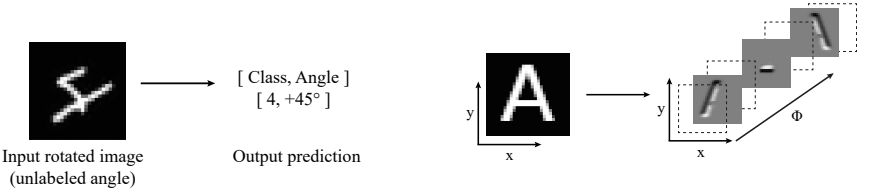
1 Introduction

Many industrial and real-life applications (e.g autonomous robot vision, amateur videos analysis) involve uncontrolled data acquisition conditions. Therefore, rotation invariance is a desired property of image classification tasks. Also, the capability of angular prediction of the rotated input would significantly help to understand the analysed scene.

The Convolutional Neural Networks (CNN) represent the state of the art technique for classification problems. Despite an indisputable quality of classification in most cases, the CNNs' performance is severely affected by rotation of the input pattern.

To obtain rotation invariance, one of the main approaches is the data augmentation technique [3] to force the network learn all rotations of the input. One consequence is that the model's size increases, and the second one is a higher probability of overfitting. Finally, such large models penalize the computational performances of the application. The model's size may even become an eliminating factor in mobile and embedded applications.

Recently, several authors proposed another way to obtain rotation invariance by mapping the input pattern into some particular space. Nonetheless, all these approaches still



(a) (b)

Figure 1: (a) Prediction of class and rotation angle. (b) Projection into roto-translational space.

require including rotated samples in the database to achieve a reasonable error rate. The second problem – the angular prediction – remains unsolved unless an angle-labeled dataset is available. However, most of the classification datasets do not include the angle information.

In this paper, we propose a new CNN architecture, containing steerable oriented filters at the first layer, able to capture the internal properties (spatial and angular relations) of the objects and recognize their rotated copies without the need of a rotated database. Our network, RED-NN, is able to predict the class and the angle when its rotated (Fig. 1(a)). The presented network has a self-organizing behavior between the random angle inputs and a linear output. Generally speaking, the network learns the *data structure* allowing to significantly reduce the size of the model compared to the state of the art solutions.

Filters at the first layer of common CNNs often happen to contain edge detectors in several orientations without any ordering or dependence between them [9]. The fundamental idea in this paper is to use at the feature stage a collection of oriented edge detectors, ordered by increasing angular orientation. Such collection of features forms a roto-translational space where local orientation is mapped as an additional orientation axis (Fig. 1(b)). The input rotation of a pattern is co-variant in this space with translation over this additional dimension. This roto-translation property brings interesting capabilities to the CNN architecture explored in this paper.

This paper is organized as follows: Section 2 describes existing approaches in state of the art. Section 3 makes a brief recapitulation of steerable filters [2] and demonstrates how we used it to obtain a roto-translation feature space. In Section 4 we present our CNN architecture. Finally, in Section 5 we present the evaluation and experiments done to validate the network characteristics and behavior.

2 Related work

Recent works in state of the art provide rotation invariance by encoding it in the internal filters of the neural networks. Normally, these approaches try to find a trade-off between the number of features needed to get a good prediction accuracy and network size. Invariance to transformations is achieved by Laptev *et al*. [4] by using rotated versions of the same sample and let the network to chose the right orientation.

Sifre and Mallat [6] use a group of convolutional scattering transforms to generate a set of feature extractors that is rotation-equivariant. Worral *et al*. [8] use a similar approach by generating a continuous resolution angular mapping using complex steerable filters. One drawback is that the rotation invariance is present at the feature stage and not in the predictor stage. The work of Marcos *et al*. [5] use several rotated instances of the same filter but they improve they results by using test-time data augmentation.

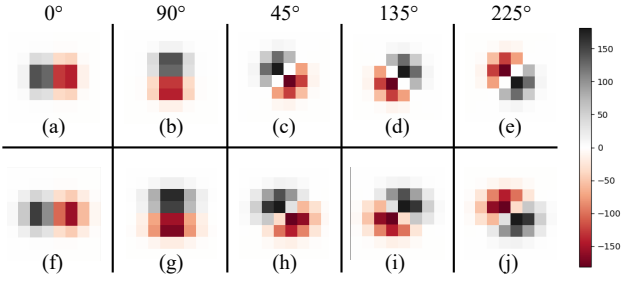


Figure 2: Steerable filters. Fig.(a-e) initial value ($\alpha, \beta, l = 1$) Fig. (f-j) filters after training ($\alpha = 0.43, \beta = 0.85, l = 1.12$). 45°, 135°, 225°are created by the linear combination of the basis filters on 0°and 90°using Eq. 5.

The filter steerability property is used by Weiler *et al*. [2] with their Steerable Filter CNN. They use linear combinations of the filters to obtain sampled orientations used to achieve rotation-equivariance. The filter banks of their implementation are rather fixed than learned. They also rely on network initialization weights to decrease the error rate off the network. The main difference between their work and us is that we are able to output a predicted angle without label and we do not apply any weight initialization technique.

For the case on training with up-right oriented samples Follman *et al*. [1] achieve rotation invariance by generating rotation invariant features. The main drawback of their approach is that they rely on increasing the size of the network (more than 1 million of parameters) to reduce their error rate. Zhou *et al*. [10] propose the Oriented Response Networks in which they have Active Rotating Filters. These filters rotate during convolution and produce maps with location and orientation explicitly encoded.

These approaches achieve good accuracy while training on randomly oriented samples but the few ones (see Table 2) that show results on up-right oriented training samples are not able to go lower than 10% of error rate. In comparison to the randomly oriented training works [1, 4, 5, 7, 8, 10] we are able to reach state of the art values while using only a fraction of the parameters. Furthermore, our architecture is able to predict the angle orientation without any angle labeling. Following the same comparison with the up-right oriented training approaches we get values as low as 1.5% of error rate with our architecture being invariant and co-variant to angular transformations of the input.

3 Steerable filters

Freeman and Adelson [2] define the term *steerable filter* as a class of filters in which a filter of arbitrary orientation is synthesized as a linear combination of a set of *basis filters*. We applied the two-dimensional case of this methodology for our architecture. Consider the two-dimensional, Gaussian function G written in Cartesian coordinates x and y:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

Substituting $l = \frac{1}{2\sigma^2}$ in the Eq. 1 we can make a generalization of this filter. Let the parameters α and β be the shape parameters on the exponent and l the scaling parameter.

$$G(x, y, l, \alpha, \beta) = \frac{l}{\pi} e^{-l(\alpha x^2 + \beta y^2)} \tag{2}$$

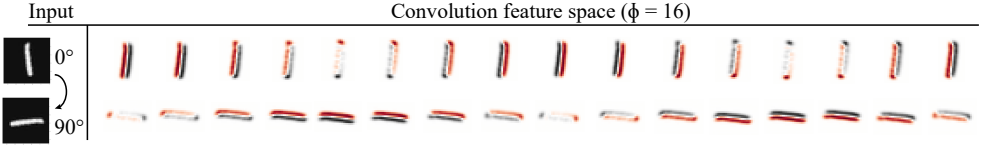


Figure 3: Convolution feature space ($\Phi = 16$) for input in up-right position (0°) and rotated (90°) before the un-rotation step. Each rotated filter $d\varphi$ is applied to the input.

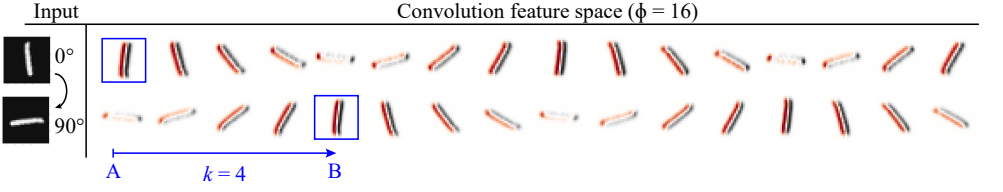


Figure 4: Convolution feature space ($\Phi = 16$) for input in up-right position (0°) and rotated (90°) after the un-rotation step. Angular input transformation is equal to $kd\varphi = 90°$.

Following Freeman's [2] methodology we calculate the first order directional derivative of Eq. 2 in the x direction. Let the n_{th} derivative of G be expressed as G_n.

$$G(x,y,l,\alpha,\beta)_1^{0^\circ} = \frac{\partial}{\partial x}\frac{l}{\pi}e^{-l(\alpha x^2+\beta y^2)} = \frac{-2\alpha l^2 x}{\pi}e^{-l(\alpha x^2+\beta y^2)} \tag{3}$$

Then the same function in y direction.

$$G(x,y,l,\alpha,\beta)_1^{90^\circ} = \frac{\partial}{\partial y}\frac{l}{\pi}e^{-l(\alpha x^2+\beta y^2)} = \frac{-2\beta l^2 y}{\pi}e^{-l(\alpha x^2+\beta y^2)} \tag{4}$$

A filter with any arbitrary orientation θ can be calculated by the linear combination of G^{0° and G^{90° using:

$$G_1^\theta = cos(\theta)G_1^{0^\circ} + sin(\theta)G_1^{90^\circ} \tag{5}$$

$G_1^{0^\circ}$ and $G_1^{90^\circ}$ are the *basis filters* and the terms (l,α,β) are shape parameter of the filter. These parameters will be optimized by training to fit the data (Fig. 2(f-j)).

Using Eq. 3(a) one *basis filter* can be mapped to a convolution filter of size n x m (Fig. 2(a)). The second one is mapped using Eq. 4 (Fig. 2(b)). The next filters can be calculated by the linear combination of the *basis filters* using Eq. 5 (Fig. 2(c-e)).

Let the number of discrete rotations Φ be the number of sampled orientations of the input. As the basis filter have odd symmetry, the rotation period will be 360°. The angular sampling steps ($d\varphi_n$) are calculated by dividing the period by the angular sampling. For example, for $\Phi = 16$ we will generate 16 filters in steps of $d\varphi = 360^\circ/16 = 22.5^\circ$. Following this methodology we can create a filter bank of size equal to the number of desired sampled orientations. Results of experiments with different values of Φ vs the accuracy are shown in Section 5.

3.1 Roto-translational feature space

The filters mapped from the scattering transform methodology applied to the input image generate as many features as number of discrete rotations Φ. An example of this output for an up-right oriented (0°) input pattern (a handwritten number "1") can be seen in Fig. 3.

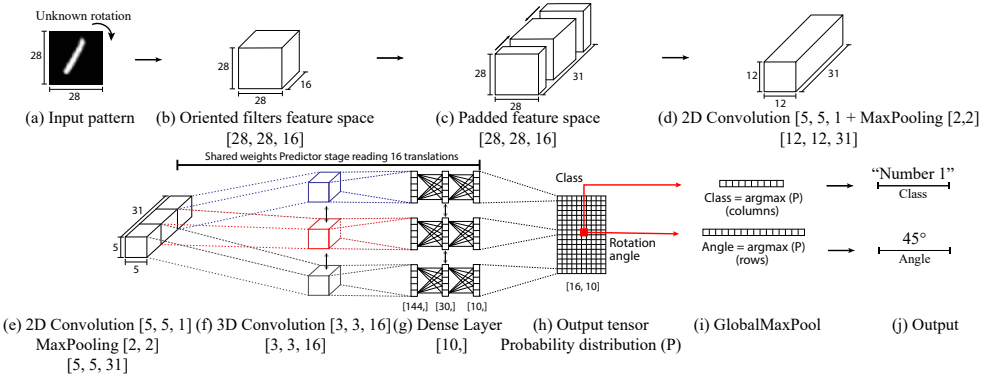


Figure 5: CNN Architecture using $\Phi = 16$ and input image size of [28, 28]. Output size of each layer between brackets. Upper-side is the feature extraction. Lower-side shows the shared predictor stage.

The rotating versions of the edge detector are applied to the input and every filter generates higher magnitude output when aligned with the input edges. We can observe that a change of 90° in the input does not generate an invariant feature translation with respect to height and width of the images.

To obtain roto-translation co-variance we compensate the rotation of each one of the filters by the magnitude of $d\varphi_n$ that generate them. This is done by rotating the corresponding feature by $-d\varphi_n$ using projective transforms. Fig. 4 shows the the highest activations vertically oriented after this compensation. When applying a 90° transformation to the input (Fig. 4) the feature space translates over the depth axis from the point A to B. This translation magnitude is proportional to the magnitude of the transformation. For the case of $\Phi = 16$ we observe a depth translation k of 4 steps. Multiplying $d\varphi_n = 22.5°$ by this translation outputs the magnitude of the angular transformation of the input. Hence, by this process we can validate the linear roto-translation properties of the layer. Also, is straightforward to see that the angular relationship between features is preserved as a linear distance between the filters.

A rotation in the input is equivariant with translation in this feature space. This space is periodic with the 2π period of complete rotation. A rotation on the input of magnitude equal to one angular step $d\varphi$ corresponds to a one step cyclic shift along the depth axis Φ of the feature space. Consequently, a model can learn spatial and angular relations of an input pattern using shared 3D Convolutions. That means that one trained predictor using a cyclic convolution and translating over the axis Φ can see all possible rotations of the input. However, to the best of our knowledge no cyclic convolution operator is available on GPU. It can nonetheless be implemented with a periodic padding of the roto-translational space by itself and a linear convolution along the Φ dimension.

This 3D convolutional predictor can read each one of the possible orientations over the Φ axis and generate a higher class probability for the translation that contains the up-right orientation. The adjacent translations ($\pm d\varphi$ from up-right translation) will infer the same class albeit with a lower probability.

4 Rotation-Equivariant Deep NN Architecture

Likewise classical CNN for classification our architecture contains a feature and predictor stage. The feature extraction stage (Fig. 5(a-d)) contains the roto-translational feature space generation and padding operation. The second stage (Fig. 5(e-j)) contains the 3D Convolu-

tional predictor stage that scans over each one of the translations and generates a probability distribution output. Results of this architecture are presented in Section 5.

Roto-translation steerable layer. This first layer reads the input image with size [height, width](Fig. 5(a)). This layer will preserve the size of the image and will output as many filters as Φ. For example, if we select $\Phi = 16$ the output of the layer will be in the shape [height, width, Φ] (Fig. 5(b)). As outlined on Section 3.1 this output has a roto-translation behavior respect to the input and the images angle compensated as discussed previously.

Periodic padding. The feature space from the previous layer is padded periodically. The output of this layer will be an padded roto-translational feature space with size [height, width, $2\Phi - 1$] containing all the possible translations of the input for the given Φ (Fig. 5(c)).

Convolutions and Maxpooling. A series of convolutional predictors are applied to the padded feature space. This predictors follow the state of the art implementations of the combination between convolution and maxpooling. To preserve the translation properties of the padded feature space these predictors are only applied to the height and width dimensions (Fig. 5(d)). They are implemented as 3D convolutions with value 1 on the translation space axis (Fig. 5(e)).

Translation scanning convolution. A 3D convolution layer is applied to the resulting feature space. This convolution is designed as a translation predictor that scans over the space with shared weights. The predictor reads each of the possible translations and generates stronger activation maps when it is applied to the up-right orientation position (Fig. 5(f)). The output contains Φ feature maps with different magnitude for each translation. Bigger activations are found on the translation corresponding to the up-right orientation.

Dense layer. The resulting features are flattened on the height, width and feature channels while preserving the Φ values of the translations. A hidden dense layer with shared weights is applied to each one of these Φ and the output vector for each translation has the length of the number of classes N (Fig. 5(g)).

The output of this stage has the form of [Φ, N] and is co-variant to the angular transformations of the input (Fig. 5(h)). This is demonstrated as the highest value in the row proportional to the angular input and on the column corresponding to the class. Also, high values probability values are observed in the adjacent rows of the output.

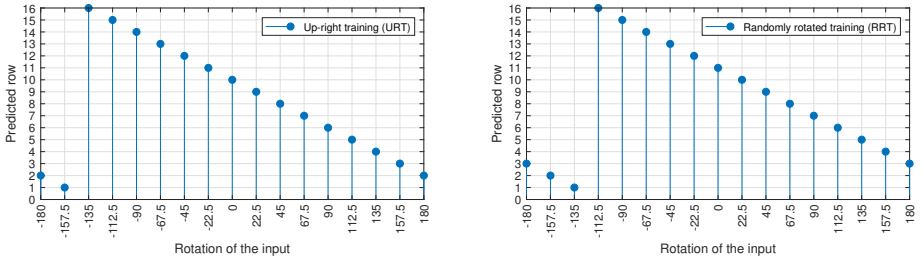
Global MaxPooling. Rotation invariance is obtained by applying a global maxpooling operation to the table formed by [Φ, N] (Fig. 5(i)). The output contains a probability distribution in the format one-out-of-many with size equal to the number of classes N (Fig. 5(j)).

5 Approach evaluation and validation

We evaluate the co-variant and invariant properties of the network on three variations of the MNIST dataset. MNIST dataset contains 60,000 training samples and 10,000 test samples. The angular sampling is $\Phi = 16$ for the main tests and one test with different Φ values is presented. The training is done by using the class labels in a one-out-of-many format. Any angular labels are provided to the network in the training phase.

Up-right training (URT) This is, to train the network as usually with up-right oriented samples, and obtain a rotational invariant prediction. We train the network with the original up-right 60,000 numbers. The validation set of 10,000 samples is rotated by an unknown random angle between $[0, 2\pi]$.

Randomly oriented training (ROT) To show that training on randomly oriented datasets and predict the rotation is possible without angular labeling available. All of the training and validation samples of the original MNIST are rotated by an unknown random angle between



(a) (b)

Figure 6: Co-variant relation between input rotation and output rows. The input (number 2) is rotated in $d\varphi$ steps. (a) Up-right training. (b) Randomly rotated training.

$[0, 2\pi]$.

Rotated MNIST dataset (rotMNIST) Rotated MNIST is usually used by state of the art approaches. To allow a direct comparison tests using this dataset are presented. This dataset contains 12,000 training samples and 58,000 testing samples randomly rotated between $[0, 2\pi]$.

5.1 Rotation angle prediction

We obtain the angular prediction from the table with shape [Φ, N]. An argmax operation on this table outputs the highest probability of orientation. To validate the behavior we did tests by rotating the same input sample in multiples of Φ.

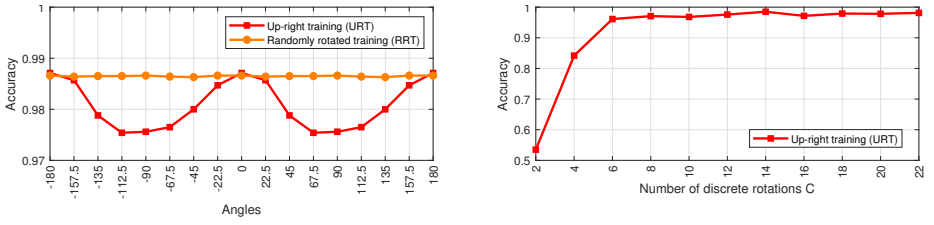
Fig. 6 shows the co-variance relation between the angular deformation in the input and the argmax of the rows at the output. A self-organizing behavior is observed in up-right and randomly rotated training cases. This comes as a result of non-zero probabilities on the neighbour translations ($d\varphi_{n+1}$ and $d\varphi_{n-1}$). Is important to notice that for the second case (Fig. 6(b)) the up-right orientation (corresponding to the vertical reference) does not exist explicitly but the network randomly selects one of the translations of the feature space as a virtual up-right position and maps the consecutive angles in an ordered way. We obtain similar results training with the rotated MNIST dataset. In cases which no vertical reference exist the rotation values are obtained by the modulo of this random virtual up-right position.

5.2 Rotation-invariant classification

Rotation-invariance property makes the output of the network invariant to angular rotations of the input image. To validate the capabilities of the architecture for different transformations of the input we test over different input angles and with different magnitudes of Φ. Also, we present a direct comparison with existing error rate values from the state of the art using the rotated MNIST dataset.

5.2.1 Network properties

Rotation-invariance test. Most of state of the art approaches validate their implementation with rotated oriented samples. To test the rotation invariance we test the accuracy on different validation sets containing each one 10,000 samples in the same orientation. We select the orientation to be in multiples of the angular sampling ($d\varphi$). As observed on the Fig. 7(a) the randomly rotated training presents a constant behavior over all the angles while the up-right trained has a slight oscillation between 97.5% and 98.7%.This slight oscillation can



(a) (b)

Figure 7: (a) Rotation-invariance test. (b) Effect of number of discrete rotations

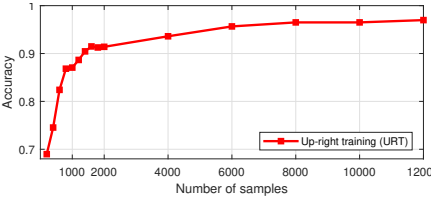


Figure 8: Effect of the training sample size test

be described as a periodic behavior from the high probability of the up-right orientations and the lower in the 90°orientation. Tests with different orientations than $d\varphi$ show the same behavior.

Angular sampling Φ ***tests.*** One of the particular parameters we present is the angular sampling Φ. We change the value of Φ from 2 to 24 to show the correlation between Φ and accuracy (Fig. 7(b)). Having more angles than 2 or 4 compared to some of the state of the art architectures is clearly helpful for the accuracy of the model. Also, we can observe that the accuracy does not longer significantly increase for values over $\Phi = 14$.

Training size test. Training size is normally proportional to the accuracy of a network. We test the architecture with different sizes of training up-right oriented samples. We observe a direct correlation between the samples and the accuracy (Fig. 8). Also, an accuracy over 90% is observed from size 1400 and bigger. Values bigger 10,000 were not plotted because the accuracy difference changes were negligible.

5.2.2 Comparison with state of the art

The architecture reaches state of the art values using rotated MNIST dataset. This comparison is made directly over the communicated error rate of each state of the art approach. We couldn't compare using other metrics because the architecture, hyper-parameters or the available dataset are not given.

Table 1 shows the comparison with state of the art implementations trained with rotMNIST dataset. We present results on the rotMNIST and our randomly rotated dataset. Is important to mention that most of this implementations have higher parameter requirements or used another techniques to improve their accuracy. Also, any of these implementations focus on the co-variance shown previously as they output only invariant class prediction in their results. While rotating the training set can be nearly catalogued as data augmentation we also present results on up-right training cases.

Table 2 presents the comparison between our implementation and the ones founded in the literature that train on up-right oriented samples. Furthermore, some implementations like (RP_RF_1_32) have over 1 million parameters. Our implementation size has 42,141 parameters with only 1 set of basis filters.

Table 1: Obtained error rate (training/validation=rot-MNIST)

Method	Error rate	# parameters
Harmonic Networks [8]	1.69%	116k
TI-Pooling [4]	1.26%	n.c.
Rotation Eq. Vector field networks[5]	1.09%	100k
ORN [10]	1.37%	397k
SFCNNs [7]	0.714%	n.c.
RP_RF_1* [1]	3.51%	130k
RED-NN [ROT] (Ours)	2.20%	42k
RED-NN [rotMNIST] (Ours)	1.39%	42k

Table 2: Obtained error rate (Up-right training URT)

Method	Error rate	# parameters
ORN-8(ORPooling)[10]	16.67%	397k
ORN-8(ORAlign)[10]	16.24%	969k
RotInv Conv. (RP_RF_1) [1]	19.85%	130k
RotInv Conv. (RP_RF_1_32)* [1]	12.20%	1M
RED-NN (Ours)	2.05%	42k

6 Conclusions

In this paper we present a trainable steerable layer to obtain a CNN architecture with rotation-invariant class prediction and rotation angle prediction capabilities. The class prediction is invariant to angular transformations of the input. The angular prediction is co-variant to the same transformations. We present several tests to demonstrate the properties of this network.

We validate the rotation invariance property of the network by demonstrating prediction on rotated inputs while only being trained on up-right oriented samples. Also, rotation co-variance tests show a self-organizing behavior on up-right and randomly oriented training samples. Hence, the predicted angle is accurate up to modulo of some arbitrary reference chosen by the network. We have presented results that reach state of the art implementation for randomly oriented training samples and outperformed existing implementations based on up-right oriented training samples.

All these properties open the possibility to several applications. One of these possible applications is the alignment of randomly oriented datasets by using the angular prediction. Other applications include those that need to be trained on un-rotated samples and obtain a rotation invariant prediction on the inference. Furthermore, the low size network in term of the number of parameters (Table 1, 2) allows to have all these capabilities on embedded or portable devices. Some tests on these devices are presented in the supplementary material.

Our model can learn the internal structure of data by the roto-translation property of the presented layer based on steerable filters. This layer transforms an input angular difference as a distance along a new axis. Rotation information is preserved along the predictor stage by presenting all the rotated copies of the feature space to a shared weights predictor scanning over them. As a consequence of these properties in the filter and predictor stages a smaller model is enough to achieve high accuracy.

Furthermore, a lower number of discrete rotations (Φ) will force the model become rotation-invariant up to $\pm d\varphi$. We have obtained results over 90% accuracy by using values $\Phi > 5$ (Fig. 7(b)).

This paper presents results with a feature space that consist of organized rotated copies of one basis filter. One future extension consists of using features based on two or more basis filters to enrich the descriptive capability of the model. Furthermore, validation on more complex datasets is expected in the near future.

References

[1] P. Follmann and T. Bottger. A rotationally-invariant convolution module by feature map back-rotation. In *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, volume 2018-Janua, pages 784–792. IEEE, mar 2018. ISBN 9781538648865. doi: 10.1109/WACV.2018.00091.

[2] W. T. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. ISSN 01628828. doi: 10.1109/34.93808.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] D. Laptev, N. Savinov, J. M Buhmann, and M. Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in Convolutional Neural Networks. 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.38.

[5] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. Rotation Equivariant Vector Field Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 5058–5067, 2017. ISBN 9781538610329. doi: 10.1109/ICCV.2017.540.

[6] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1233–1240, 2013. doi: 10.1109/CVPR.2013.163.

[7] M. Weiler, F. A Hamprecht, and M. Storath. Learning Steerable Filters for Rotation Equivariant CNNs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00095.

[8] D. E Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic deep: Networks translation and rotation equivariance. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 7168–7177, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.758.

[9] M. D Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8689 LNCS, pages 818–833, 2014. ISBN 9783319105895. doi: 10.1007/978-3-319-10590-1_53.

[10] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 4961–4970, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.527.