

A Minimal Model for Classification of Rotated Objects with Prediction of the Angle of Rotation

Rosemberg Rodriguez Salas, Petr Dokládal, Eva Dokladalova

▶ **To cite this version:**

Rosemberg Rodriguez Salas, Petr Dokládal, Eva Dokladalova. A Minimal Model for Classification of Rotated Objects with Prediction of the Angle of Rotation. 2020. hal-03118567v1

HAL Id: hal-03118567

https://minesparis-psl.hal.science/hal-03118567v1

Preprint submitted on 22 Jan 2021 (v1), last revised 18 Feb 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Minimal Model for Classification of Rotated Objects with Prediction of the Angle of Rotation

Rosemberg Rodriguez Salas[a,*], Petr Dokládal[b], Eva Dokladalova[a]

[a]*LIGM, University Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France*
[b]*MINES Paris – PSL Research University, Center for Mathematical Morphology, 35 rue Saint Honoré, Fontainebleau, France*

Abstract

In classification tasks, the robustness against various image transformations remains a crucial property of the Convolutional Neural Networks (CNNs). It can be acquired using the data augmentation. It comes, however, at the price of the risk of overfitting and a considerable increase in training time. Consequently, other ways to endow CNN with invariance to various transformations – and mainly to the rotations – is an intensive field of study.

This paper presents a new reduced rotation invariant classification model composed of two parts: a feature representation mapping and a classifier. We provide an insight into the principle and we prove that the proposed model is trainable. This model is smaller in terms of trainable parameters than similar approaches, and has angular prediction capabilities.

We illustrate the results on the MNIST and CIFAR-10 datasets. On MNIST, we i) achieve the state of the art of classification on MNIST-rot (with training on MNIST-rot), and ii) improve the results of classification on MNIST-rot (with training on upright MNIST). When trained on CIFAR-10 with upright samples and tested with rotated samples we improve by 20% the state of the art classification results. In all cases, we can predict the rotation angle.

Keywords:
2000 MSC: ,,, Image Classification, Convolutional Neural Network, Rotation Invariance, Prediction of Angle of Rotation, Steerable Filters

1. Introduction

Frequently, images and videos are acquired in uncontrolled conditions in the industry (e.g. autonomous navigation systems, aerial imagery, video surveillance) and in real-life applications (amateur photos and videos). Thus, the applied image classification method has to be robust w.r.t. variations of scale, rotation, shear, and other deformations. In this paper, we focus on the robustness of classification w.r.t. the rotation. We can cite several automated systems where the object rotates arbitrarily, e.g. food recognition [1], where no upright orientation exists, as the classification of galaxies [2, 3], aerial imagery [4] or in biology (cells in microscope images). Also, the estimation of the object's rotation angle is needed in robotics to estimate the pose as for e.g. grasping objects [5] or in other real-life applications [6, 7, 8].

Recall the terms equivariance, invariance and covariance of a function $f(\cdot)$ with respect to a transformation $g(\cdot)$, as defined in [9]:

- equivariance: $f(g(\cdot)) = g(f(\cdot))$,
- invariance: $f(g(\cdot)) = f(\cdot)$,

*Corresponding author: email: r.rodriguez@esiee.fr

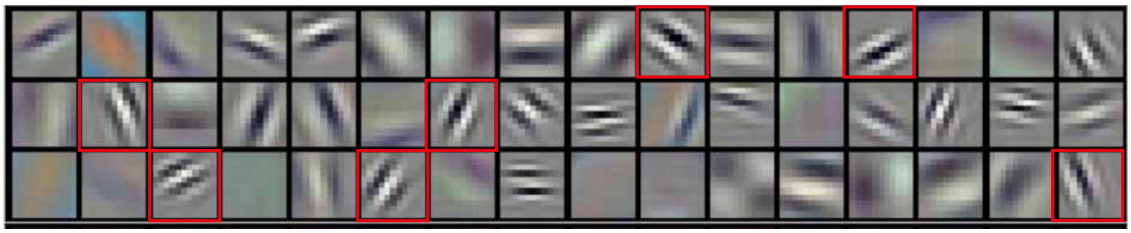


Figure 1: The first layer of AlexNet architecture [16]. Red boxes show an example of rotated versions of almost the same filters.

- covariance: $f(g(\cdot)) = g'(f(\cdot))$,

where g' is another transformation, which is itself a function of $g(\cdot)$. With the above definitions, equivariance and invariance are special cases of covariance. Hence, the classification of a randomly rotated object is invariant to rotation, and prediction of the angle of the rotation is covariant with the rotation.

Today, the CNN represent the state of the art for classification tasks. It is well known that one of the main CNN properties is the translation equivariance of the feature representation learned in the first convolutional layers. On the contrary, the rotation invariance of the classification has only recently drawn attention and the literature rapidly grew abundant.

Classically, the CNN deal with rotations by using data augmentation [10]. This means a network is trained with a dataset enriched by rotated, mirrored, sheared and scaled original images. In this way, each class becomes a super-class containing all possible deformations of itself. The main consequence is that the classifier itself is forced to become invariant to all these deformations.

However, the data augmentation comes at the price of a higher probability of overfitting and a significant increase in training time [11]. Also, certain data augmentation techniques are not safe when used on rotation transformation, adding errors in the labels for some complex datasets [12]. One possible solution to avoid label transformations is to use the dataset in its original orientation. A rotation-invariant network can provide the prediction on randomly rotated images while being trained on the original orientation (commonly upright) and labels. This approach allows to obtain smaller networks with the ability to predict the rotation angle.

1.1. Contribution

In this paper, we present a new network organization with the possibility of rotation-invariant prediction of the class and unsupervised prediction of rotation. Notice that the network preserves equally the translation invariance for the rotated images. We propose a Rotation Equivariant Deep Neural Network (RED-NN) containing - in the first layer - a rotating filter, as seen in some state-of-the-art approaches [13, 14, 15]. Instead of learning - in the first layer - a number of filters that are potentially, randomly-rotated copies of each other (Fig. 1), we learn a unique basis filter, and generate a set of its rotated copies.

The learned filters and associated activations serve to capture the inner geometric properties of the input example. These properties represents the mutual spatial and angular relationships between the edges of the input example. The global orientation is suppressed and the example can be represented in arbitrary orientation. Then a collection of predictors poll the representations of the input example in different orientations and predict a class probability. The position of the maximum probability corresponds to the angle of rotation of the input example.

We validate our approach on the MNIST and the CIFAR-10 data sets to compare objectively with the related published results. In the case of MNIST, we achieve the state-of-the-art results with a single basic filter and a much smaller model. In the case of CIFAR-10, we improve the results related in the literature by 20% of accuracy, again with a much smaller model.

1.2. Paper organization

This paper is organized as follows. Section 2 contains a review of related works on rotation-invariant classification. Section 3 introduces the theoretical background of the proposed method and proves that the network is trainable. Section 4 recalls the principles and equations of steerable filters, and explains the roto-translational properties of our filter ensemble. Section 5 describes the proposed architecture, and

Table 1: State of the art in the domain of rotation-invariant object classification: summary of main properties. Legend: Size : Model size when applied to MNIST dataset; Upright - upright oriented examples only; n.c. - non communicated.

		Model		Output		Training approach	
Name	Dataset	Size	Tested rotations	Classification	Angle	Upright	Randomly rotated
TI-Pooling [17]	MNIST-rot	n.c.	24	yes	no	no	yes
Harmonic networks [18]	MNIST-rot	33k	Continuous	yes	no	no	yes
Spherical CNN [19]	MNIST/MNIST-rot	68k	Continuous	yes	no	yes	yes
SFCNNs [20]	MNIST-rot	n.c.	24	yes	no	yes	yes
GCNs [21]	MNIST-rot CIFAR10	1.86M	4	yes	no	no	yes
Icosahedral CNN [22]	MNIST/MNIST-rot	n.c.	n.c.	yes	no	yes	yes
RotEqNet [9]	MNIST/MNIST-rot	100k	17	yes	no	yes	yes
RI-LBCNNs [23]	MNIST-rot	390k	8	yes	no	no	yes
ORN-8 [24]	MNIST/MNIST-rot CIFAR10	969k	8	yes	no	yes	yes
Rot.-Inv. Conv. [25]	MNIST/MNIST-rot CIFAR10/CIFAR10-rot	11M	8	yes	no	yes	yes
Covariant CNN [14]	MNIST/MNIST-rot	7k	16	yes	yes	yes	yes
RED-CNN (this paper)	MNIST/MNIST-rot CIFAR10/CIFAR10-rot	42k	16	yes	yes	yes	yes

Section 6 explains the layer characteristics of our implementation. The experimental results are collected in Section 7. It contains different experiments validating the network's capability of invariant learning and comparisons with concurrent methods. The conclusions bring the outline of the presented achievements and propose some perspectives of future extensions.

2. Related work

The literature contains numerous references on endowing the CNNs with rotation invariance or equivariance (see [9] for definition). Two main strategies exist: i) data augmentation or ii) modification of the network architecture. The second one can be further divided following the domain of the features [18] and how they are processed: group convolutions, continuous rotation angle sampling (for 360°rotation equivariance) and discrete angle sampling for rotating filters.

2.1. Data augmentation

Historically, the first technique used to significantly improve the invariance to the rotation of the CNNs were data augmentation techniques [10]. The basic principle is to enrich the training data with the artificially transformed images. This technique results in learning generalized models without modifying the network architecture. The major disadvantage of this approach is the need to increase the computing capacity for learning while increasing the overfitting risk [20]. Another documented disadvantage is the unsafe training due to label transformations when the dataset is augmented. This label noise is a product of the automatic pre-processing of the dataset (i.e., zooming in a section of the image that does not contain the class object). Also, when the dataset naturally includes images that are not class invariant when rotated, the label does not update automatically, which can result in a low prediction accuracy [12].

We can include in this category approaches using the rotated images at the input of the network. In this sense, we can cite here TI-POOLING [17]. The authors use rotated versions of the same image as the input while letting the network choose the right orientation thanks to integrating TI-POOLING operators and parallel Siamese architectures.

To avoid the need for dataset and label transformations, and to work with smaller datasets, the CNN development effort tends to propose the networks allowing training with only upright samples. However, it requires to modify the CNNs internal structure.

2.2. Internal architecture modification

For several years, the trend converges towards encoding the rotation equivariance directly in the CNN architectures. Usually, these works use a bank of oriented filters, rotated by a finite number of angles. Depending on how the authors generate the filter bank, we can divide these approaches into group convolutions, continuous domain angle sampling (in hard-baked 360°rotation equivariance) approaches based on the work of Mallat et al. [18], and a finite number of filters generated from a single mother wavelet which is rotated using Steerable filters techniques [26].

Group convolutions. In general, group convolution approaches find a trade-off between the number of sampled discrete orientations and the size of the network. They usually use pooling techniques at the end of the group convolutions. As a result, the equivariance capacity of the network is diluted into a single rotation invariant result. The main consequence is a rotation invariant prediction and the loss of the equivariant information in the network.

In this context, Dieleman at al. [3] represents one of the first significant works. It relies on deep symmetry networks. The authors introduce four operations (roll, stack, pool, slice) that can be inserted into neural network models as layers, making their models partially equivariant to rotations. They also integrate parameter sharing across different orientations, obtaining smaller models.

We can also mention other, previous publications such as the Multi-Column Deep Neural Networks [27]. They train a model for each one of the possible transformations, followed by a global averaging pooling. Gens and Domingos [28] propose a generalization of CNNs that forms a component of feature transformation maps over arbitrary symmetry groups. In the last, the Spatial Transformer Network [29] applies spatial transformations to the feature maps.

To improve the rotation equivariance capability of CNNs, Follman et al. [25] present the Rotationally-Invariant Convolution Module with rotational convolutions and rotational pooling layers. They achieve rotational invariance by rotating the filters and then back-rotating the generated feature map by the filter's negative angle. They present results also the results on CIFAR-10 dataset trained on upright samples and validated on rotated ones. Notice that there exist a similar approach published earlier, Polar Transformer Network [30]. It transforms the input to polar coordinates with the origin learned as the centroid of a single channel.

Marcos at al. [9] introduce a CNN architecture encoding rotation equivariance, called Rotation Equivariant Vector Field Networks (RotEqNet). The network applies one filter at different orientations and extracts a vector field feature map, encoding the maximum activation in terms of magnitude and angle. The best results, despite test time data augmentation, achieve relatively high error rate (20%) when trained on upright and validated on randomly rotated samples.

Recently, Li et al. [31] presented the Deep Rotation Equivariant Network. Their network is based on ResNet34 [32] and specific up-sampling and projection layers encoding the rotation and reflection symmetry of dermoscopy images. However, they do not achieve a complete rotation equivariance.

In RotDCF [33], the authors propose a decomposition of the filters in group equivariant CNNs; they show benefits in reducing the parameters and computational complexity. They also illustrate how the decomposition of the convolution filter across the 2D space leads to an implicit regularization of the filters and improves the robustness of the learned representations. However, they only present results on rotated validation up to 60°while training on upright samples.

Continuous rotation angle sampling. This group allows moving from discrete orientations to continuous sampling. The principle has been introduced by the harmonic networks (or H-Nets) in Worall et al. [18]. H-Nets exhibit equivariance to patch-wise translation and continuous 360-rotation. It obtains this behavior by replacing the regular CNN filters with circular harmonics, returning a maximal response and the orientation for every receptive field patch.

There are several similar approaches based on the H-nets theory. We can cite the Spherical CNN [19] and Icosahedral CNN [22]. The first one uses a spherical cross-correlation, which is expressive and rotation-equivariant. This network also allows training with upright position samples only; however, the computational cost remains very high.

Icosahedral CNN [22] proposes an improvement to the Spherical CNN. Mainly, the authors pay attention to the reduction of computational cost. They proceed by sub-sampling the sphere in a single 2D Convolution call and make it scalable. A similar approach is represented by CubeNet [34]. It uses a 3D

CNN group convolution that permutes the output as a function of the input while keeping the time 2× slower than non-group CNNs. However, they need the rotational data augmentation because of the lack of angular selectivity.

Discrete angle sampling for rotating filters. Here, the main principle is to generate an oriented filter bank by using some filter rotation technique (i.e., the Steerable filters [26]). Usually, the network generates during the learning process a set of mother wavelets and then rotates them to generate oriented responses.

One of the first proposal has been published by Zhou et al. [24]. The authors propose the Oriented Response Networks in which they have Active Rotating Filters. These filters rotate during the convolution and produce maps with location and orientation explicitly encoded. The main drawback is high number of parameters to learn. Next, we can cite Rotation Invariant Local Binary Networks [23]. The authors introduce the Local Binary Orientation Module. It can be inserted into a traditional CNN. This layer contains Local Binary Convolutional and Active Rotating Filters.

A significant contribution is recent work of Weiler et al. [20]. Their SFCNN comprises a set of complex filters and rotates them by phase manipulation. However, these filters have learned weights (the magnitude of the activation changes but the shape remains constant). While this work is similar to ours, their results are presented only on the classification of the rotated MNIST dataset. Compared to this work, we use a single real mother wavelet rotated using the Steerable Filters technique [26]. We let the filters learn weights, shape and size parameters. This allows the filters to change the shape and size during the training. We also provide our network with the angular prediction capability relying on the relative position of the filters in the network. Furthermore, SFCNN uses a generalization of He's weight initialization to improve their results while our network is independent of weight initialization.

Recently, the idea of learning steerable filters inspired the introduction of layers based on the Gabor filter bank [21]. They also present interesting results using the ResNet as backbone (instead of convolutional predictor) of their network for Natural Image Classification tasks. Whereas the obtained accuracy is very competitive, the authors present the results only for four different orientations, and the number of parameters increases to almost 2 million parameters.

In conclusion, the methods using various versions of orientable filters represent the state of the art approaches when dealing with rotation invariant networks. One of the main disadvantages of these methods is that calculating the complex filters increases the network's training time and size. The general approach is to have learnable weights but there are only few papers proposing to learn the shape and size parameters. Also, according to our knowledge, there is a real lack of networks allowing to predict the rotation angle. Our work permits to show that the state of the art accuracy can be achieved with only a single real mother filter reducing radically the size of the model.

To complete, we can also cite Zhong et al. [35] discussing deeply the theoretical aspects of invariance/equivariance transformations.

To conclude, our analysis of the state of the art shows that the models transforming the input image have considerable larger training times and are prone to overfitting than those based on the rotation invariance (or equivariance) built in the network. It complies with the extensive study of Aquino [36], McGuiness [11] and Quiroda [37]. The current state of the art CNN achieve the rotation invariance by average pooling from the output of the network but loses the input's angular information. In contrast, in this paper we preserve the equivariance information to predict the angle of the input.

Table 1 summarizes the properties of the selected state-of-the-art methods in terms of the number of orientations, the model size, and the capability predicting the angle of the input rotation. The selection of the analyzed methods depends on the accuracy when training a network with upright samples and testing with randomly rotated samples. We believe that only this demonstrates that the network learns efficiently to classify rotated samples it has never seen during the training phase.

In this paper, we introduce a modified network architecture predicting the rotation angle through a layer of ordered, shared-weights predictors scanning all possible discrete rotations of the input. We obtain a class-invariant inner mapping where the translation is equivariant with the rotation of the example. Our workflow allows to train the network with upright samples avoiding label transformations problems generated on data augmentation approaches. The model acquires the capability to learn the angle of rotation in a non-supervised way. All the predictors use the output of a unique representation mapping and share their weights. As a consequence, the overall size of the model remains small. We detail these principles in the following section.

3. Methodology

This section introduces the theoretical background of the proposed method and proves that the network is trainable. First, we start by introducing the feature space representation.

3.1. Feature space

Let $\mathbf{x}$ be an example (an image) $\mathbf{x} \in \mathbb{R}^{m\times n}$, and let g be a filter acting as an oriented edge detector with the periodicity 2π. Let ρ_φ be a transformation rotating the support by the angle φ. Then $\rho_\varphi g = g^\varphi$, with $g{=}g^{0^o}$, be such a filter oriented along the angle φ. See below, eq. 12, for definition of such a filter used in this paper.

The product $\mathbf{x} * g^\varphi$ extracts the oriented components of $\mathbf{x}$ that are oriented in φ. We have a set of orientations $\varphi_i = 2\pi i/N$ for $i = 0, \ldots, N$, with $N \in \mathbb{Z}^+$, and $d\varphi = 2\pi/N$. The ordered set $[\mathbf{x} * g^{\varphi_i}]$ contains all oriented components of $\mathbf{x}$.

Definition 1 (***feature representation).*** *Let* Φ *be a mapping* $\Phi : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n\times N}$*, and* $\Phi(\mathbf{x})$ *a feature representation of* $\mathbf{x}$

$$\Phi(\mathbf{x}) = [\rho_{-\varphi_i}(\mathbf{x} * g^{\varphi_i})] \tag{1}$$

containing oriented components of $\mathbf{x}$*, and unrotated to align with the referential orientation* φ_0*.*

Consider now a special type of permutation τ where the elements move rightwards and the last one replaces the first one. This permutation is cyclic with the period equal to the length of the vector it is applied to. We refer to τ by *translation*, and τ_i denotes τ applied i-times. In this context, the translation is applied along the third dimension. That is, applied to $\mathbf{x}$, Φ and $[g^{\varphi_i}]$, in this context, the following holds:

1. A translation τ applied to some $\mathbf{x}$ is the identity operator $\tau\mathbf{x} = \mathbf{x}$ because $\mathbf{x}$ lacks the third dimension.
2. $\tau[g^{\varphi_i}] = [\rho_{d\varphi} g^{\varphi_i}]$ because of $\rho_{d\varphi} g^{\varphi_i} = g^{\varphi_{i+1}}$
3. $\tau_k[g^{\varphi_i}] = [\rho_\vartheta g^{\varphi_i}]$ for some $\vartheta = kd\varphi$, $\forall k \in Z$

We have the following property:

Property 1. *There is a covariance of the rotation of* $\mathbf{x}$ *with the translation of* $\Phi(\mathbf{x})$ *along its third axis*

$$\tau_k\Phi(\mathbf{x}) = \Phi(\rho_\vartheta \mathbf{x}) \quad with \quad \vartheta = kd\varphi, \quad \forall k \in \mathbb{Z} \tag{2}$$

Proof 1. *We have*

$$\begin{aligned}
\Phi(\rho_\vartheta \mathbf{x}) &= [\rho_{-\varphi_i}(\rho_\vartheta \mathbf{x} * g^{\varphi_i})] \\
&= [\rho_{-\varphi_i}\rho_\vartheta(\mathbf{x} * \rho_{-\vartheta} g^{\varphi_i})] \\
&= [\rho_{\vartheta-\varphi_i}(\mathbf{x} * g^{\varphi_i-\vartheta})] \\
&= \tau_k[\rho_{-\varphi_i}(\mathbf{x} * g^{\varphi_i})] = \tau_k\Phi(\mathbf{x})
\end{aligned}$$

Considering all $k = 0, \ldots, N$ the left-hand side of eq. (2) gives access to features coresponding to all rotations $\rho_\vartheta \mathbf{x}$. Among all these rotations, there is one that corresponds to an almost unrotated version of $\mathbf{x}$.

3.2. Prediction model

For some given, joint probablity $p(y, x)$ let us search for a model assigning to x a class y. However, not x but only $\rho_\vartheta x$ is observable. A classical way of learning a model to approximate p is training the model using a data augmentation $y = f^{\mathrm{DA}}(\rho_\vartheta x)$. Using a feedforward network we learn a mapping

$$y = f^{\mathrm{DA}}(\rho_\vartheta x; \boldsymbol{\theta}) \tag{3}$$

where $\boldsymbol{\theta}$ are the model parameters and the superscript DA indicates a data augmented model. We typically proceed by maximizing the probability of the prediction by minimizing a cost function $J(\boldsymbol{\theta}) = -\log p(y \mid \rho_\vartheta x)$.

Instead of that classical way, consider a model f in the form of concatenation of two mappings $f(\cdot) = f^*(\Phi(\cdot))$, where Φ is a representation mapping and f^* a class-probability model. Using the mapping Φ from eq. 1 and considering all its permutations $\tau_i\Phi$ for $i = 1, \ldots, N$ we obtain a collection of models

$$[f_i(\cdot)] = [f^*(\tau_i\Phi(\cdot))] \tag{4}$$

with $f^*(\cdot) = f^*(\cdot, \boldsymbol{\theta}_{f^*})$ and $\Phi(\cdot) = \Phi(\cdot, \boldsymbol{\theta}_\Phi)$. The parameters $\boldsymbol{\theta}_\Phi$ and $\boldsymbol{\theta}_{f^*}$ are independent of the rotation of x. We analyze this collection of models below.

In the following the term $p(A)$ denotes the probability of some stochastic event A to occur. More particularly, when $p(A)$=1, it indicates the certainty even though A is stochastic.

Definition 2 (*Indiscernibility*). *Let f be a classification model, $\mathbf{x}_1$ and $\mathbf{x}_2$, with $\mathbf{x}_1 \neq \mathbf{x}_2$, two examples and $f(\mathbf{x}_1)$ and $f(\mathbf{x}_2)$ the predictions by f on these examples. When f predicts the same class for any $\mathbf{x}_1$ and $\mathbf{x}_2$, that is $p(f(\mathbf{x}_1)=f(\mathbf{x}_2)) = 1$ then we say $\mathbf{x}_1$ and $\mathbf{x}_2$ are indiscernible by f, written $\mathbf{x}_1 \simeq_f \mathbf{x}_2$.*

Property 2 (*Equivalence of representation*). *Let f be some classification model in the form of $f(\cdot) = f^*(\Phi(\cdot))$, and $\rho_\psi\mathbf{x}_1$ and $\rho_\varphi\mathbf{x}_2$ be two differently oriented examples. If the two different, and differently oriented, examples are indiscernible by the model, that is $p(f(\rho_\psi\mathbf{x}_1)=f(\rho_\varphi\mathbf{x}_2)) = 1$ then we write*

$$p(f^*(\tau_i\Phi(\rho_\psi\mathbf{x}_1)) = f^*(\tau_j\Phi(\rho_\varphi\mathbf{x}_2))) = 1$$

after rotation of the support by $\rho_{-\psi}$ we obtain

$$p(f^*(\tau_i\Phi(\mathbf{x}_1)) = f^*(\tau_j\Phi(\rho_{\varphi-\psi}\mathbf{x}_2))) = 1$$

substitution $\vartheta=\varphi-\psi$

$$p(f^*(\tau_i\Phi(\mathbf{x}_1)) = f^*(\tau_j\Phi(\rho_\vartheta\mathbf{x}_2))) = 1$$

using the Definition 2 to drop the probability

$$\tau_i\Phi(\mathbf{x}_1) \simeq_{f^*} \tau_j\Phi(\rho_\vartheta\mathbf{x}_2)$$

by translating both sides by τ_{-j} we finally obtain the indiscernibility of representation of some rotated example $\mathbf{x}_2$ and a translated representation of some upright $\mathbf{x}_1$.

$$\exists k, \quad \tau_k\Phi(\mathbf{x}_1) \simeq_{f^*} \Phi(\rho_\vartheta\mathbf{x}_2), \qquad\qquad \textit{with } k = i-j$$

There is a way of finding a convenient k for some given (and unknown) ϑ by using the maximum likelihood.

Lemma 1. *For any $\vartheta \in \mathbb{R}$, and $0 < d\varphi$ sufficiently small*

$$\exists k \textit{ such that } \tau_{-k}\Phi(\rho_\vartheta\mathbf{x}) \simeq \Phi(\mathbf{x}) \tag{5}$$

and k depends on $\mathbf{x}$.

Proof 2. *From Prop. 1 for $\vartheta=d\varphi$ and $k=1$ we have the equality $\tau_{-k}\Phi(\mathbf{x}) = \Phi(\rho_\vartheta\mathbf{x})$. However, because $\vartheta \in \mathbb{R}$ but $k \in \mathbb{Z}^+$ and when $kd\varphi \neq \vartheta$, $\forall k$, the exact equality is not possible and only $\tau_k\Phi(\rho_\vartheta\mathbf{x}) \simeq \Phi(\mathbf{x})$ since a misalignment occurs between $\Phi^{-1}(\tau_{-k}\Phi(\rho_\vartheta\mathbf{x}))$ and $\Phi(\mathbf{x})$ but at most up to $\pm\frac{d\varphi}{2}$.*

Because of the Lemma 1, one of the models in eq. 4 will maximize the class probability predicted from each representation mapping $\tau_i\Phi$

$$\max_i p_{f_i}(y = \mathbf{y} \mid \rho\mathbf{x}) \tag{6}$$

We want to minimize the negative log-likelihood cost of the model providing the correct class prediction

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\log p(y = f^*(\tau_k \Phi(\rho_\vartheta x) \mid x, \vartheta)) \\ \text{where } k &= \arg\max_i p(y = f^*(\tau_i \Phi(\rho_\vartheta x) \mid x, \vartheta)) \end{aligned} \tag{7}$$

in a usual way by minimizing the per-example loss

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = -\log p(y \mid x, \vartheta; \boldsymbol{\theta})$$

When the k-th model maximizes the probability in eq. 7 we adapt the weights of the k-th model by taking $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon g$ where $g = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\rho_\vartheta x, y; \boldsymbol{\theta})$. The weights of layers are updated by using the error back propagation algorithm as usual $f_k(.; \boldsymbol{\theta}) = f^*(\tau_i \Phi(.; \boldsymbol{\theta}_\Phi); \boldsymbol{\theta}_{f^*})$. Notice that ϑ in $\rho_\vartheta x$ is not needed to be known since the weights $\boldsymbol{\theta}_\Phi$ and $\boldsymbol{\theta}_{f^*}$ are independent of ϑ. Updating the weights of the mapping $\tau_k \Phi \mathbf{x}$ is done using $\tau_k[\rho_{-\varphi_i}(\mathbf{x} * g^{\varphi_i})]$ which consists of three functions: translation τ_k, a set of fixed-angle rotations $\rho_{-\varphi_i}$ and convolution of x by a set of filters g^{φ_i}, where only the parameters of the edge detector g are trainable. This training is a classical closed-loop training iterated until convergence.

3.3. The Vapnik-Chervonenkis Dimension of the model

Recall the Vapnik-Chervonenkis (VC) dimension of a classifier, defined as the maximal number of different points $\mathbf{x}$ that the model can label arbitrarily. Intuitively, a more complex problem requires a model with a higher VC dimension, and the model size will be bigger.

Consider now a model f assigning a class y to x for some joint probability $p(y, x)$.

When only $\rho_\vartheta x$ are observable, the classification $\hat{y} = f(\rho_\vartheta x)$ can be done by using the data augmentation to train $y = f(\rho_\vartheta x)$, with all ϑ, see eq. 3. This model data will have a high VC dimension.

Consider now a f fitted to $p(y, x)$, i.e. with no data augmentation. When only $\rho\mathbf{x}$ is observable, the classification $\hat{y} = f(\rho_\vartheta \mathbf{x})$ and the prediction of the angle ϑ could also be done by using the maximum likelihood polling, eq. 6. This approach however would be computationally costly since the prediction needs to be done over all the rotations.

If the model f can be split into two parts, as in eq. 4, the predictions $f(\rho_{\vartheta_i} \mathbf{x})$, for all possible ϑ_i, become very cheap. We compute a unique feature representation $\Phi(\mathbf{x})$. The predictions are then computed by the second part of the model f^* on the mere translations $\tau_i \Phi(\mathbf{x})$.

The VC dimension of f, eq. 4, is much smaller than that of f^{DA}, eq. 3, since f learns the representation of $p(y, \rho_\vartheta x)$ with only $|\vartheta| < \frac{d\varphi}{2}$ instead of $\vartheta \in (0, 2\pi)$ when data augmentation is used. Consequently, the model size also is smaller.

Moreover, for the model f, eq. 4, we have $|\boldsymbol{\theta}_f| = |\boldsymbol{\theta}_{f^*}| + |\boldsymbol{\theta}_\Phi|$, where $|\cdot|$ denotes the number of parameters. The mapping $\Phi(\cdot)$ is computed only once, and the predictions $[f_i^*(\cdot)]$ are cheap since using only a subset of parameters $|\boldsymbol{\theta}_{f^*}|$.

4. Steerable filters

Theoretically, the feature space representation can be built around any filter that has the faculty of being oriented in a particular direction.

In this paper, we decide to use the steerable orientable filters. The orientable filters have been studied by Freeman and Adelson [26]. They define the term *steerable filter* as a class of filters in which a filter of arbitrary orientation is a linear combination of a set of *basis filters*.

We apply the two-dimensional case of this methodology as follows. Consider the two-dimensional, Gaussian function G written in Cartesian coordinates x and y:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{8}$$

By substituting $l = \frac{1}{2\sigma^2}$ in eq. 8 we obtain the general expression in eq. 9. α and β are the shape parameters and l is the scaling parameter.

$$G(x, y, l, \alpha, \beta) = \frac{l}{\pi} e^{-l(\alpha x^2 + \beta y^2)} \tag{9}$$

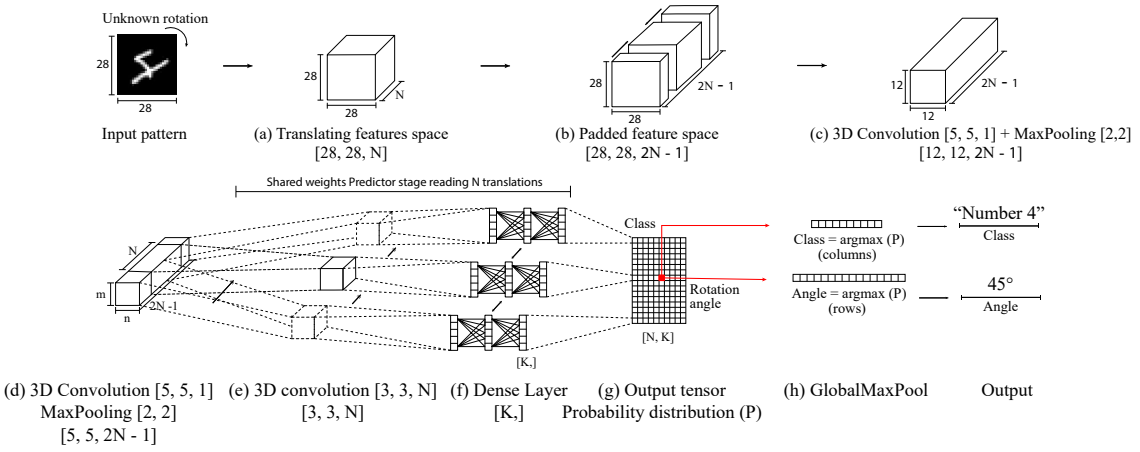


Figure 2: CNN Architecture using $N = 16$ and input image size of [28, 28]. The number of classes $K = 10$. Output size of each layer between brackets.

Following Freeman's [26] methodology we calculate the first-order directional derivative of eq. 9 in the x direction. Let the derivative of G according to x be denoted by $g(...)^{0^\circ}$

$$g(x, y, l, \alpha, \beta)^{0^\circ} = \frac{\partial}{\partial x} \frac{l}{\pi} e^{-l(\alpha x^2 + \beta y^2)} = \frac{-2\alpha l^2 x}{\pi} e^{-l(\alpha x^2 + \beta y^2)} \tag{10}$$

The derivative of G in y direction gives $g(...)^{90^\circ}$

$$g(x, y, l, \alpha, \beta)^{90^\circ} = \frac{\partial}{\partial y} \frac{l}{\pi} e^{-l(\alpha x^2 + \beta y^2)} = \frac{-2\beta l^2 y}{\pi} e^{-l(\alpha x^2 + \beta y^2)} \tag{11}$$

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

$$g^{\varphi} = cos(\varphi) g^{0^\circ} + sin(\varphi) g^{90^\circ} \tag{12}$$

g^{0° and g^{90° are the *basis filters* and the terms (l, α, β) are the shape parameters of the filter.

Then we can create the bank of N filters g^{φ_i} each oriented along the angle φ_i as introduced in the first paragraph of subsection Feature space (3.1). In addition, these filters are ordered with the increasing orientation angle φ_i.

5. Rotation-Equivariant CNN Architecture

Following the presented methodology, we construct our rotation-translation equivariant CNN architecture. The feature representation building stage is illustrated in Fig. 2(a-d) and it obeys eq. 1. Then, the predictor stage is outlined in Fig. 2(e-f). It relies on a translating predictor that scans over all the translations and generates a probability distribution output (eq. 4 - eq. 7).

Feature representation construction: by using the oriented filter bank in the first layer, we start the construction of the translating feature space (Fig. 2(a)). Then, a 3D Convolution and MaxPooling operation in Fig. 2(d) outputs the feature space representation Φ. Notice we need to guarantee that a rotation in the input image covaries with the translation at the output of this stage (Property 1). Hence the implementation has to consider the following aspects:

1. **Periodic padding.** To implement correctly the convolutions on the border of the translating feature space, we would need to consider cyclic border properties of convolutions. To obtain such behavior using linear convolutions, we pad symmetrically the translating feature space along the φ dimension (Fig. 2(b)). Then we use a linear 3D (x, y, φ) convolution sliding along the mentioned φ dimension. The output of this layer will be a padded translational feature space [height, width, 2N–1] containing all the possible translations of the input for the given φ_i.



(a) Train airplane 0° (b) Train dog 0° (c) Test bird 225° (d) Test car 30°

Figure 3: Upright training (a,b) and Randomly rotated (c, d) examples. All examples are processed to avoid black corners and borders. Test examples are rotated randomly between 0° and 360° clockwise.

2. **Convolutions and MaxPooling.** To reduce the size of the feature space prior to applying a predictor, we apply a series of convolutions and max pooling, Fig. 2(c-d). They are implemented as 3D convolutions with the value 1 on the translation space axis to have the same properties as a cyclic convolution over the axis N. All the convolutional predictors are followed by a ReLu activation.

Predictor. All the models share the same predictor f^* applied to $\tau_i\Phi$, $i = 1, \ldots, N$ (eq. 4). We implement the predictor using a 3D convolution layer Fig. 2(e) followed by a dense layer Fig. 2(f).

1. **3D convolution.** The convolution slides over the rotation axis φ to sequentially read all the rotations. The output contains N feature maps with different magnitude for each translation.
2. **Dense layer.** The 3D convolution output is flattened on the height, width, and feature channels while preserving the N values of the translations. A hidden dense layer with shared weights is applied to each position φ_i. The output vector for each translation has the length of the number of classes K.

The output of this stage, Fig. 2(g), is a probability distribution in the form of $[N, K]$ with N the number of discrete orientations, and K the number of classes. The highest probability is found in the row corresponding to the orientation of the input and on the column corresponding to the class (Lemma 1).

The class prediction is obtained by applying a global MaxPooling operation to the table formed by $[N, K]$ (Fig. 2(h)). The output contains a probability distribution in the one-out-of-many format with the size equal to the number of classes K (Fig. 2(h)).

To improve the convergence, we insert a batch normalization layer to avoid the vanishing gradient problem and we apply a dropout layer with a value of 0.5 to regularize and avoid overfitting. We use the ReLU activation for the hidden layer and a softmax activation for the output neurons.

6. Training

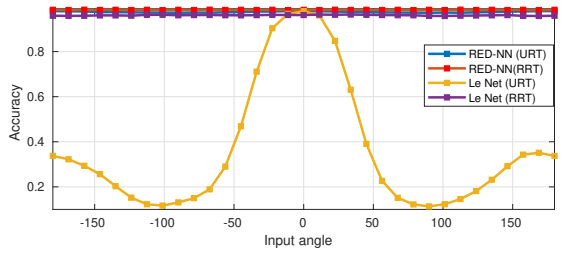
In eq. 4, the convolutional predictor f^* scans all $\tau_i\Phi$ (corresponding to all rotations $\rho_\vartheta x$) and outputs the predictions of class and angle for x. It predicts a high-class probability on the translation that corresponds to the seemingly upright rotation of the example. Retrieving this position gives the possibility of predicting the rotation angle.

To obtain the class and the rotation angle we pool the row and column indexes of the maximum of the probability distribution Fig. 2(g). The column index of the maximum is computed using the GlobalMaxPool Layer from the Tensorflow framework. The row index is also computed using the GlobalMaxPool Layer after transposing the probability map.

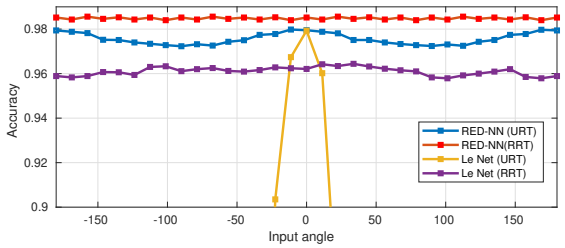
Also, as we need the steerable edge detectors to be adapted to the training set, the parameters α, β, l need to be trainable weights of the network. These weight parameters become part of the neural network graph and update on each iteration of the training cycle. The initial value of these parameters are: $\alpha = 0.5, \beta = 0.5, l = 0.5$.

The angular prediction occurs only during the prediction. For the training, we disconnect the transpose and angular GlobalMaxPooling.

The GlobalMaxPool in the classes is present during the training. Its presence during the training plays a crucial role in being able to predict the angle of the inference phase. The input image (a 2D array) and the class ground-truth (one-hot encoding) are used during the training, but no angular information is needed. The class ground-truth is backpropagated during the GlobalMaxPool layer towards the position where the maximum probability class prediction occurred and zeros elsewhere. In this way, the prediction of the row at the seemingly upright position is reinforced, whereas the misaligned ones are diminished. (Recall that all

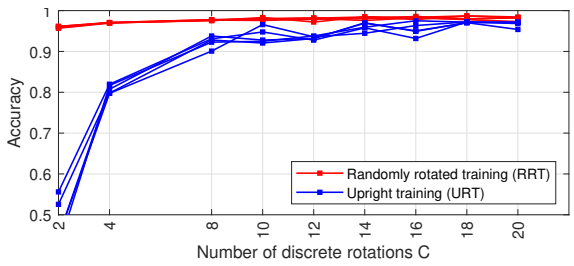


(a) RED-NN vs conventional CNN accuracy (URT)

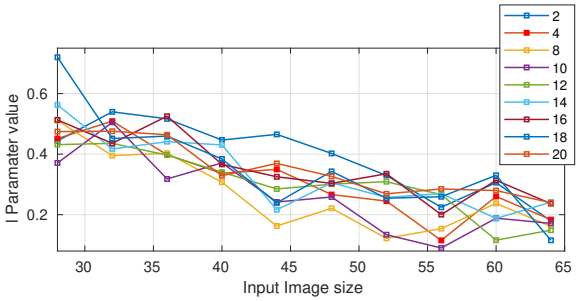


(b) RED-NN vs conventional CNN accuracy (RRT)

Figure 4: **Accuracy comparison.** (a) Conventional CNN's trained with upright samples (URT) do not possess the rotation invariance. (b) Even vs conventional CNN's with randomly rotated training (RRT) our network performs better in terms of accuracy.



(a) Effect of N on accuracy for URT and RRT



(b) Effect of N on learned value l.

Figure 5: (a) With upright training the prediction accuracy slightly decreases for non-orthogonal orientations. (b) Parameter l learned value is inversely proportional to the size of the input pattern. This behavior is consistent for $N = 2$ to 20.

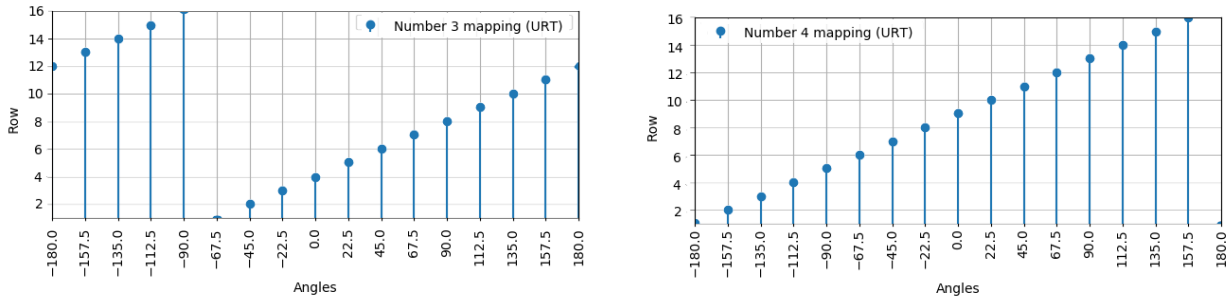
the predictors share their weights.) The next time the same class is presented to the network (albeit rotated differently), the prediction of this class is reinforced again, even though the example is rotated differently.

Given the angular sampling dφ then it suffices for the dense layer classifier be rotation invariant up to $\pm$d$\varphi/2$. A finer angular sampling dφ will require from the classifier a smaller rotation invariance, hence a smaller model. Notice that a self-organizing behavior appears in the form of a continuous mapping of the rotation angle to the probability distribution P, see Fig. 2(g). This is a consequence of the design of the network and the training. First, the predictors Fig. 2(e-f) are ordered at an increasing angle. Second, during the training, an example pattern, rotated in an angle not necessarily precisely equal to an entire multiple of $d\varphi$, is presented to the network on training. This makes the classifier become rotation-invariant to misalignments up to $\pm$d$\varphi/2$. When during the training, a pattern is presented to the network, then one column of the architecture will predict a maximum class probability but also its immediate neighbors will predict the same class with a somewhat smaller probability of the same class. Continuous mapping of the rotation progressively appears during the training. This behavior is similar to that of the Kohonen self-organizing maps [38] without being explicitly fostered algorithmically.

For experiment results, we train the network with upright oriented samples and randomly oriented samples following the same methodology for both. The results of this architecture are presented in Section 7.

7. Experiments

In this section, we present the classification results on the rotated MNIST and CIFAR-10 datasets. We test both training possibilities, on the upright-oriented and on a randomly-rotated dataset. We compare our results with the state of the art. We analyze the network capabilities by showing results on different sizes of images, different magnitudes of N, and the impact of the size of the training set. Also, we present results that validate how the learning parameters of the layer change for different size of input images. At



(a) Number 3 mapping (URT)

(b) Number 4 mapping (URT)

Figure 6: **Self-organizing mapping behavior** The network consecutively maps the angles to each one of the rows of the output table. It selects a virtual upright position for each one of the classes. (a) For the class "number 3" the virtual upright is 4, (b) For the class "number 4" the virtual upright is 9.

the end of this section, we present the angular prediction capacity of the architecture, and we show the self-organizing behavior of the prediction.

7.1. Experimental settings

For all the tests, the same network architecture was used. For comparison with literature, we used values published in other papers. We ran the experiments on a GTX Titan X GPU with 12 GB RAM.

Angular sampling N=16 is used for all unless noted differently. We use the same hyperparameters in all the tests, Adam algorithm for the optimizer, Dropout rate (0.5) for the dense layers, batch size 64, and trained the network for 50 epochs for each test. The implementation is based on Tensorflow 1.13.1.

7.2. Datasets

Upright training (URT) The MNIST is a toy dataset for experimenting network performance. It contains a distribution of 60,000 training and 10,000 validation samples. In this type of training, we validate the rotation invariance of the network. We keep the training set in the original upright position and the validation set is rotated by a random angle between $[0, 2\pi]$.

Rotated MNIST dataset (MNIST-rot) The rotated MNIST is usually used by the state of the art approaches. To allow a direct comparison with the literature we present tests on this dataset. This dataset contains 12,000 training samples, and 58,000 testing samples randomly rotated between $[0, 2\pi]$.

CIFAR-10 To further test the network capabilities we use the CIFAR-10 dataset. It contains 10 classes of images captured in different conditions. We train the network with upright samples and validate on randomly rotated samples. Following the approach proposed by Follman and Bottger [25] we process each example to avoid artifacts when rotating them; the images are cropped by a circle with radius of half the image size and the black border is smoothed by a Gaussian with a kernel size of five pixels.

7.3. Prediction of the angle of rotation

We obtain the angular prediction from the probability distribution P with the shape [N, K]. The column index of the maximum provides the predicted class. The row index multiplied by $d\varphi$ gives the rotation angle w.r.t. the vertical reference.

Since no vertical reference is provided in the training phase, the network randomly selects a virtual upright position V_i for each one of the classes. The consecutive angles are mapped in an ordered way, see Fig. 6, conformally to Prop. 1. Note, that this virtual upright position is random and different for each class of the dataset.

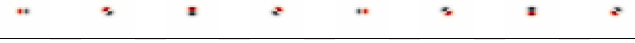
Input image size	Learned filter ensemble
28 x 28 px (a)	
64 x 64 px (b)	
80 x 80 px (c)	

Figure 7: **Filter ensemble for different input size** The learned parameter l controls the size of the filter. Larger input images results in an increased size of the filter. (a) $l = 0.51$ (b) $l = 0.16$ (c) $l = 0.09$. $N = 8$ for the three cases.

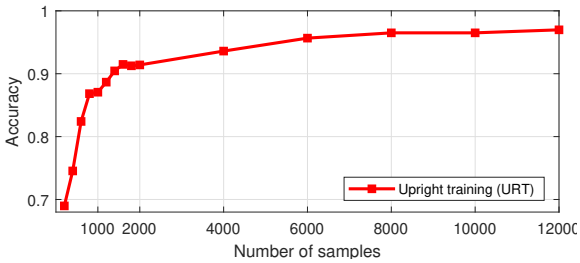


Figure 8: **Training size vs accuracy.** The number of samples heavily affects the accuracy up to 15,000 samples.

7.4. Rotation-invariant classification

To validate the invariance of the prediction under the rotation of the input, we test different input rotations with different values of N. We provide a direct comparison with existing state-of-the-art error-rate values on the rotated MNIST dataset (MNIST-rot).

Rotation-invariance test. Most of the 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 examples in the same orientation. We select the orientation to be in multiples of the angular sampling ($d\varphi$). As observed in Fig. 4(a), the randomly rotated training presents a constant accuracy over all the angles while the upright training has a slight oscillation between 97.5% and 98.7% Fig. 4(b). This slight oscillation can be described as a periodic behavior from the high probability of the upright orientations and the lower in the 90°orientation. Tests with orientations different from $d\varphi$ show similar behavior. For illustration purposes, the LeNet model, trained with upright oriented samples, is not able to recognize the rotated versions of the input, and the accuracy falls off when the rotation increases. When the LeNet is trained with randomly rotated samples (data augmentation), the accuracy does not exceed the accuracy of our network.

Angular sampling N tests. One of the particular parameters we test is the number of tested rotations N. We change the value of N from 2 to 24 to show the correlation between N and accuracy (Fig. 5(a)). Having $N > 4$ (as most of the state-of-the-art architectures) is helpful for the accuracy of the model. Also, we can observe that the accuracy does not significantly increases for values over $N = 14$ – both the upright and the randomly oriented trainings present this behavior.

Training size test. The training size is normally proportional to the accuracy of a network. We test the architecture with different sizes of training upright oriented samples. We observe a direct correlation between the samples and the accuracy; see Fig. 8. Also, when evaluated for size 1400 and larger, we obtain an accuracy over 90%. Values bigger than 10,000 were not plotted because the accuracy difference changes were negligible.

Training parameter l test. The l parameter of the basis filters defines the width of the edge detector filter. Lower values of l indicate a wider edge detector.

Fig. 5(b) shows the correlation between the input size and the l value. For larger input images, the value of l decreases, and the filters grow larger. The change in the filters can be observed in Fig. 7. This proves the ability of the network to adapt the edge detector to the size of the input. This behavior is consistent for different values of N demonstrating the training capability of this parameter.

Table 2: Obtained error rate (Training/Validation=MNIST-rot)

Method	Error rate	# parameters
Harmonic Networks [18]	1.69%	33k
TI-Pooling [17]	1.26%	n.c.
R. Eq. VFN[9]	1.09%	100k
ORN [24]	1.37%	397k
SFCNNs [20]	0.714%	3.3M [39] *
RP_RF_1* [25]	3.51%	130k
RI-LBCNNs [40]	1.36%	390k
GCNs [21]	1.10%	1.86M
Covariant CNN[14]	2.69%	7k
RED-NN (this paper)	0.93%	42k

Table 3: Obtained error rate (Training=URT/Validation=MNIST-rot)

Method	Error rate	# parameters
ORN-8(ORPooling)[24]	16.67%	397k
ORN-8(ORAlign)[24]	16.24%	969k
(RP_RF_1) [25]	19.85%	130k
(RP_RF_1_32)* [25]	12.20%	1M
RotDCF (60 degrees) [33]	17.64%	760k
Spherical CNN [19]	6.00%	68k
Icosahedral CNN [22]	30.01%	n.c.
RI-LBCNNs [40]	25.77%	390k
Covariant CNN[14]	17.21%	7k
RED-NN (this paper)	2.05%	42k

Table 4: Obtained error rate (Training=CIFAR-10/Validation=CIFAR-10 rotated)

Method	Error rate	# parameters
RP_RF_1 [25]	55.88%	130k
ORN-8[24]	59.31%	382k
RP_1234 [25]	62.55%	130k
RED-NN (this paper)	36.41%	73k

7.5. Comparison with state of the art

The architecture reaches the state of the art values using the rotated MNIST dataset, and outperforms state of the art implementation on the CIFAR-10 upright training and rotated validation. This comparison is made directly over the communicated error rate of each state of the art approach. We could not compare using other metrics because the architecture, hyper-parameters, or the available dataset are not always given.

Table 2 shows the comparison with state of the art implementations trained and validated with the MNIST-rot dataset.

When using $N = 16$, we observe that we keep the number of parameters low compared to other implementations. Harmonic Networks also have a lower number of parameters but at the price of increased computational power needed to disentangle their harmonic frequencies.

The accuracy outperforms the previous state of the art implementations when using the MNIST-rot dataset. We can see that SFCNN achieves lower values in terms of error rate, but at the cost of high number of parameters. Also, SFCNN has a bank of fixed complex filters that have learnable weights. In contrast, our filters have trainable parameters and weights. Having trainable parameters allows the filter to change in shape and size during the training, as shown in Fig. 8 and demonstrated in Section 7(b).

Table 3 presents the comparison of existing works and ours on the upright orientated training and

randomly rotated samples. The weakness of the networks to generalize on previously unseen examples (differently rotated) makes an impact on the error, see Table 2.

We outperform the present state of the art techniques with our network. While Spherical CNN achieves low values, it is at the price of high complexity and processing power.

In terms of parameters, the proposed network has lower parameters than the other works. The value of N can lower the number of parameters. Oriented Response Networks also present results with fewer orientations (ORN-4) obtaining 59.67% loss. In Fig. 7(a), we show that with 4 orientations, we achieve 18% loss, making our network more robust against changes in sampled orientations.

Our previous work [14] has the lowest number of parameters but the error rate is high due to the non-trainable scattering transform used to generate the roto-translational space.

SFCNN error rate of 0.714% is comparable to this paper. Recently, Graham et al. [39] trained and disclosed SFCNN number of parameters of 3.3M. We achieve similar results (less than 0.2% error rate) using 42k parameters. This considerable difference comes from using a single mother wavelet and learning all the filter parameters. In addition, recall that our network uses the relative position of the translation on the roto-translational feature space to predict the input's angle.

Table 4 compares our results with other implementations that use upright samples on the training and rotated for the validation. We can observe that our approach outperforms existing state of the art implementations with only a half (56.15%) of trainable parameters. The reported results on Table 4 are with 8 orientations. We obtain similar results for higher number of orientations (36.00% for N=12, 35.81% for N= 16). Similar to the results obtained on the MNIST dataset, Fig. 5(a), increasing furthermore the number of orientations ($N > 16$) did not increase the accuracy.

All these results are obtained using a single learnable basis filter rotated N times. Also, we keep the equivariant properties of the filters allowing to predict the angular transformation of the input. In this sense, this is the first approach using the roto-translational properties to predict the angle.

7.6. Application: Automatic orientation of datasets

Several datasets in the literature contain randomly oriented samples of each one of the classes. Manually orienting them by labeling the angle can be time-consuming. Angular prediction opens the possibility of the automatic alignment of randomly oriented datasets. The automatic orientation of the datasets application aligns each one of the samples to a vertical reference by using the predicted angle of the network (Fig. 9).

Our network can predict the angular difference of the input images with respect to the vertical. For this application, we rotate the input image by the negative of the predicted angle. As a result, we obtain a vertically oriented input image.

It is important to notice that for this application, we trained the network with an upright oriented dataset and a randomly oriented dataset and obtained satisfactory results for both cases. For the case of randomly rotated training, a vertical reference was not provided. The network selects some virtual upright position randomly and then maps the consecutive angles following this reference. This property allows to orient the randomly oriented samples to the virtual upright position chosen by the network.

7.7. Embedded devices implementation

Wearable devices like cell phones or embedded solutions using single-board computers (Raspberry Pi, Beaglebone and others) have various constraints on available resources. The low memory footprint of our network allows the implementation in these type of devices.

We implemented the network in a cell phone (Samsung S8 with GPU) and a single-board computer (Raspberry Pi 3B). It demonstrates the possibility to implement our network conforming to CPU limitations and memory constraints. Furthermore, the prediction time required by the network was not heavily impacted compared to the typical implementation. For the case of the Raspberry Pi, the prediction time is about 80 ms, and the portable phone is 35 ms compared to 20 ms and 10 ms of the original LeNet implementation. We can observe a faster time on the cellphone due to the GPU accelerator it has. To compare, on a CPU the processing time is in the order of milliseconds. These timings include the classification and rotation prediction of the input. A demonstration video of both of these implementations is available in [41] and the source code in [42].

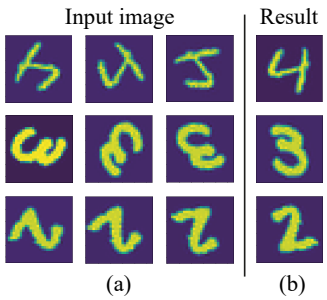


Figure 9: **Automatic alignment of numbers** (a) Randomly oriented input samples. (b) Upright oriented results using the angle prediction of the network.

8. Conclusion

In this work, we present a CNN architecture capable of classifying rotated images and predicting the angle of rotation. We validate the capability to classify randomly rotated images after training the model in two ways: 1) on upright oriented examples and 2) on randomly oriented examples without providing the orientation angle.

The model outputs the angle of rotation, which is a discrete value corresponding to the orientation in which the maximum class probability occurs over N tested orientations. We obtain over 90% class prediction accuracy for already very few tested orientations (for $N \geq 8$). A higher accuracy, reaching or surpassing state of the art, can be obtained by increasing N to obtain a finer angular sampling $\mathrm{d}\varphi$. When more rotations are tested (a larger N), a finer angular sampling (a smaller $\mathrm{d}\varphi$) allows the predictor to be rotation-invariant to smaller misalignment – typically $\pm\mathrm{d}\varphi$. This is beneficial for better prediction accuracy that is possibly obtained with a smaller predictor.

The rotation-invariant class prediction and detection of the rotation angle open the possibility of usage in several applications, e.g. automatic alignment and classification of randomly oriented datasets, classification in applications with uncontrolled orientation angle or in applications with training on upright samples but requiring rotation invariant inference.

The presented results are based on a single first-gaussian-derivative basis filter. Future extensions include several possibilities: i) using multiple learnable filters with different parameter values, or ii) using filters from some other, more flexible family, like e.g. the Gabor wavelets, where the angular and frequency bandwidth can be set independently, iii) combination with more advanced architectures (e.g. the ResNet, DenseNet, VGG as in [43, 44, 45]) as backbone instead of the presented convolutional predictor. In the last, iv) when classifying rotated objects, it might be profitable to use a custom loss, such as the Pixels-Intersection-over-Union (PIoU) [46]. The idea is that an oriented bounded box – a one aligned with a rotated object – presents less overlap with the background in complex environments.

Acknowledgment

The authors thank Christian Lantuéjoul from MINES Paris – PSL Research University for a useful discussion.

References

[1] G. Ciocca, P. Napoletano, R. Schettini, Food recognition: A new dataset, experiments, and results, IEEE Journal of Biomedical and Health Informatics PP (2016) 1–1.

[2] C. Lintott, K. Schawinski, S. Bamford, A. Slosar, K. Land, D. Thomas, E. Edmondson, K. Masters, R. C. Nichol, M. J. Raddick, A. Szalay, D. Andreescu, P. Murray, J. Vandenberg, Galaxy Zoo 1: data release of morphological classifications for nearly 900 000 galaxies, Monthly Notices of the Royal Astronomical Society 410 (2011) 166–178.

[3] S. Dieleman, J. D. Fauw, K. Kavukcuoglu, Exploiting cyclic symmetry in convolutional neural networks, CoRR abs/1602.02660 (2016).

[4] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, L. Zhang, DOTA: A large-scale dataset for object detection in aerial images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3974–3983.

[5] I. Shin, S. Nam, H. Yu, R. Roberts, S. Moon, Conveyor visual tracking using robot vision, in: Florida Conference on Recent Advances in Robotics, FCRAR, 2006, pp. 1–5.

[6] Y. Zhang, Z. Lu, J.-H. Xue, Q. Liao, Multiresolution approach to automatic detection of spherical particles from electron cryomicroscopy images, in: IEEE International Conference on Multimedia and Expo (ICME), , IEEE, 2019, pp. 846–850.

[7] Y. Zhang, Z. Lu, J.-H. Xue, Q. Liao, A new rotation-invariant deep network for 3d object recognition, in: IEEE International Conference on Multimedia and Expo (ICME), 2019, pp. 1606–1611.

[8] L. H. . Z. B. . S. X. . Z. Yongting, Cnn-based model for pose detection of industrial pcb, in: International Conference in Intelligent Computation Technology and Automation, ICICTA, IEEE, 2017, pp. 390 – 393.

[9] D. Marcos, M. Volpi, N. Komodakis, D. Tuia, Rotation Equivariant Vector Field Networks, in: Proceedings of the IEEE International Conference on Computer Vision, volume 2017-Octob, 2017, pp. 5058–5067. doi:10.1109/ICCV.2017.540. arXiv:1612.09346.

[10] D. A. Dyk, X. L. Meng, The art of data augmentation, Journal of Computational and Graphical Statistics 10 (2001) 1–50.

[11] S. O'Gara, K. McGuinness, Comparing data augmentation strategies for deep image classification, in: IMVIP 2019: Irish Machine Vision & Image Processing, Technological University Dublin, Dublin, Ireland, 2019. doi:10.21427/148b-ar75.

[12] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (2019) 60.

[13] D. Marcos, M. Volpi, D. Tuia, Learning rotation invariant convolutional filters for texture classification, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 2012–2017.

[14] R. Rodriguez Salas, E. Dokladalova, P. Dokladal, Rotation Invariant CNN Using Scattering Transform for Image Classification, in: 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 654–658. doi:10.1109/ICIP.2019.8804467.

[15] J. Bruna, S. Mallat, Invariant Scattering Convolution Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 1872–1886.

[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1907 – 1105.

[17] D. Laptev, N. Savinov, J. M. Buhmann, M. Pollefeys, Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 289–297.

[18] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, G. J. Brostow, Harmonic Networks: Deep translation and rotation equivariance, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7168–7177. doi:10.1109/CVPR.2017.758. arXiv:1612.04642.

[19] T. S. Cohen, M. Geiger, J. Koehler, M. Welling, Spherical CNNs, in: ICLR, 2018, pp. 1 – 15. arXiv:1801.10130.

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

[21] S. Luan, C. Chen, B. Zhang, J. Han, J. Liu, Gabor convolutional networks, IEEE Transactions on Image Processing 27 (2018) 4357–4366.

[22] T. Cohen, M. Weiler, B. Kicanaoglu, M. Welling, Gauge equivariant convolutional networks and the icosahedral CNN, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, PMLR, Long Beach, California, USA, 2019, pp. 1321–1330. URL: http://proceedings.mlr.press/v97/cohen19d.html.

[23] X. Zhang, L. Liu, Y. Xie, J. Chen, L. Wu, M. Pietikäinen, Rotation Invariant Local Binary Convolution Neural Networks, in: Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017, volume 2018-Janua, 2018, pp. 1210–1219. URL: http://www.outex.oulu.fi/. doi:10.1109/ICCVW.2017.146.

[24] Y. Zhou, Q. Ye, Q. Qiu, J. Jiao, Oriented response networks, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, volume 2017-January, 2017, pp. 4961–4970. doi:10.1109/CVPR.2017.527. arXiv:1701.01833.

[25] P. Follmann, 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, IEEE, 2018, pp. 784–792. doi:10.1109/WACV.2018.00091.

[26] W. T. Freeman, E. H. Adelson, The Design and Use of Steerable Filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (1991) 891–906.

[27] D. Cireşan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3642–3649. doi:10.1109/CVPR.2012.6248110. arXiv:1202.2745.

[28] R. Gens, P. M. Domingos, Deep symmetry networks, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 2537–2545. URL: http://papers.nips.cc/paper/5424-deep-symmetry-networks.pdf.

[29] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2017–2025. URL: http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf.

[30] C. Esteves, C. Allen-Blanchette, X. Zhou, K. Daniilidis, Polar transformer networks, in: International Conference on Learning Representations, 2018, pp. 1 – 15. URL: https://openreview.net/forum?id=HktRlUlAZ.

[31] J. Li, Z. Yang, H. Liu, D. Cai, Deep Rotation Equivariant Network, Neurocomputing 290 (2018) 26–33.

[32] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 770–778.

[33] L. Gao, H. Li, Z. Lu, G. Lin, 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, UbiComp/ISWC '19 Adjunct, ACM, New York, NY, USA, 2019, pp. 551–557. URL: http://doi.acm.org/10.1145/3341162.3349330. doi:10.1145/3341162.3349330.

[34] D. Worrall, G. Brostow, CubeNet: Equivariance to 3D Rotation and Translation, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 11209 LNCS, 2018, pp. 585–602. doi:10.1007/978-3-030-01228-1_35. arXiv:1804.04458v1.

[35] Z. W. Pan Zhong, Characterization and design of generalized convolutional neural network, in: Anual Conference in Information Sciences and Systems (CISS), IEEE, 2019, pp. 1–6.

[36] N. R. Aquino, M. Gutoski, L. T. Hattori, H. S. Lopes, The effect of data augmentation on the performance of convolutional neural networks, Braz. Soc. Comput. Intell (2017).

[37] F. Quiroga, F. Ronchetti, L. Lanzarini, A. F. Bariviera, Revisiting data augmentation for rotational invariance in convolutional neural networks, in: J. C. Ferrer-Comalat, S. Linares-Mustarós, J. M. Merigó, J. Kacprzyk (Eds.), Modelling and Simulation in Management Sciences, Springer International Publishing, Cham, 2020, pp. 127–141.

[38] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (1990) 1464–1480.

[39] S. Graham, D. Epstein, N. Rajpoot, Dense Steerable Filter CNNs for Exploiting Rotational Symmetry in Histology Images, arXiv preprint arXiv:2004.03037 (2020).

[40] C. Shin, J. Yun, Deep rotating kernel convolution neural network, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE, 2019, pp. 441–442.

[41] R. Rodriguez Salas, Rotation equivariant deep neural network, 2019. URL: https://youtu.be/GfWYZuZ4YJs.

[42] R. Rodriguez Salas, Rotation equivariant deep neural network, 2019. URL: https://github.com/red-nn/RED-NN.

[43] M. Shaha, M. Pawar, Transfer learning for image classification, in: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, 2018, pp. 656–660.

[44] R. Gopalakrishnan, Y. Chua, L. R. Iyer, Classifying neuromorphic data using a deep learning framework for image classification, in: 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), IEEE, 2018, pp. 1520–1524.

[45] Z. Chen, T. Zhang, C. Ouyang, End-to-end airplane detection using transfer learning in remote sensing images, Remote Sensing 10 (2018) 139.

[46] Z. Chen, K. Chen, W. Lin, J. See, H. Yu, Y. Ke, C. Yang, PIoU loss: Towards accurate oriented object detection in complex environments, in: European Conference on Computer Vision, Springer, 2020, pp. 195–211.